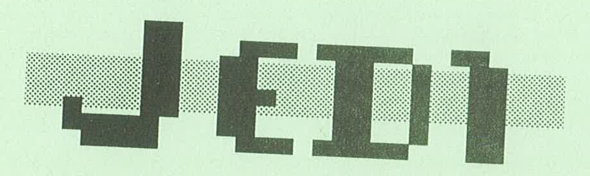
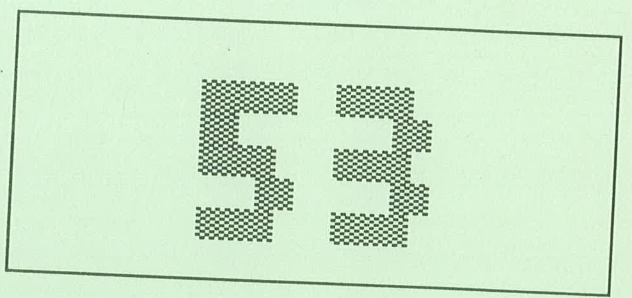


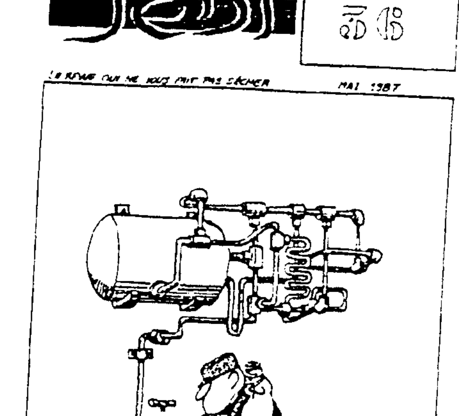


LE JOURNAL QUI NE PERD PAS SES BOULONS

AOUT 1989

SOMMAIRE DES ANCIENS NUMEROS

De 1983 à 1989, l'ASSOCIATION JEDI qui s'était fixée pour but la promotion du langage Forth, a publié un magazine nommé JEDI (Journal d'Etudes et de Documentation Informatique). Certains numéros sont totalement épuisés. Cependant, il est possible de commander les numéros dont la liste figure ci-dessous, ceci dans la limite des stocks disponibles.



Les numéros 1 à 17, 23 sont épuisés.

18 MUMPS: 5ème partie. LPB: 7ème leçon. Langage C: KERMIT.

19 FORTH: trois tours de main; opérations fractionnaires; images de grande dimension sous B3X; matrice 3D pour HRX; extensions pour Jupiter ACE; communication série pour Jupiter ACE; TONE sans vibrato pour HRX; DUMP hexa et ascii pour HRX. PILOT: un langage pour l'EAO. MUMPS: 6ème partie. COBOL: sur APPLE, suite. I.A: bases de données et syst.exp; FUTURSYS et FUTURLOG.

20 FORTH: une règle à calcul; virgule flottante F83; la vectorisation. MUMPS: 7ème partie. LPB: 8ème partie. FUTURSYS et FUTURLOG, suite.

21 FORTH: micro-traitement de texte; sculptures sonores en Forth. LISP: traitement de propositions logiques. PASCAL: fonctions graphiques pour AMSTRAD.

22 APL: démystifier APL. MUMPS: 8ème partie. FORTH: assembleur 8086 minimal; conversion AN/NA; DEFER en 79-Standard; gestion de données avec fBASE I; tracé de sinusoïdes.

24 LPB: 9ème leçon, un compilateur musical. FORTH: Expert2, une application au bâtiment; chaînage de vocabulaire en Fig-FORTH. MUMPS: la commande de boucle.

25 FORTH: les en-têtes séparés; traitement des ensembles; éditeur plein écran pour AMSTRAD; horloge temps réel pour TO7-70. MUMPS: 10ème partie.

26 MUMPS: les entrées sorties. FUTURLOG: 2ème leçon. FORTH: les variables locales; le langage Blaise, 1er épisode. LPB: 10ème leçon. APL: sur micros. PASCAL: une poignée de tools en turbo.

27 FUTURLOG et FUTURSYS. FORTH: le langage Blaise, second épisode; mini langage graphique; les micro-ordinateurs de ROCKWELL; extraction de racine; programmation des piles et des registres. MUMPS: dernière partie. PROLOG: programme d'aide au diagnostic.

28 FORTH: le langage Blaise, troisième épisode; routines générales; les mots "NONCE"; la programmation structurée; fonction sinus, tracé d'arc de cercle; résumé commandes F83. APL: une première approche. LPB: colonisation d'un IBM PC. I.A: bases de données et systèmes experts.

29 FORTH: le phrasé en Forth; exécution vectorisée; le langage Blaise, dernier épisode; utilitaires F83 pour IBM; prise de contact avec F83 CP/M et MSDOS; machine à calculer. MATHS: signature et hashcode. PROLOG: initiation, opérations arithmétiques. LANGAGE C: compteur de parenthèses.

30 FORTH: les pseudo-constantes; gestion de clavier en F83-MSDOS. LISP: LeLisp Objet. PROLOG: vérificateur de règles de dessin pour circuits intégrés en technologie CMOS. PASCAL: lien de bibliothèque avec turbo. MATHS: la compression de l'information, codage de Huffman.

31 APL: gestion de tableaux. LOGO: une gestion de fichiers. FORTH: éditeur plein écran pour AMSTRAD; fonction horloge temps réel pour IBM; fonctions graphiques pour AMSTRAD; structures de données PL/1; impression graphique multi-écran pour HRX; la gestion des fichiers en F83; sauvegarde des blocs pour AMSTRAD PCW.

32 FORTH: INPUT en F83; présentation FIG HAMBOURG; racine carrée 16/32 bits; NOVIX 4000; opérateurs d'entrée alpha-numérique; chaînes de caractères; programmation robot MULTISOFT. TELEMATIQUE: réseaux et codes TRANSPAC. APL: notion de fonctions. PASCAL: compresseur de fichiers.

33 FORTH: lancement F83 depuis dBASEII; lancement F83 depuis WORDSTAR; génération d'une commande F83 depuis une chaîne de car.; conversion + transfert fichiers texte. PROLOG: complément au prog du no30. PASCAL: décompresseur codage de Huffman, suite. SYSTEME: le moniteur du TO9+. QR: Q3 Q4 Q5 Q6 Q7 Q8 Q9. MATHS: arithmétique des entiers équilibrés.

34 FORTH: R2 R2 R5; notation algébrique infixée; assembleur 6809 1ère part.; éditeur plein écran pour F83-6809; la maitrise du graphisme. PASCAL: éditions de fichiers compressés en codage de Huffman. PROLOG: génération de grilles de LOTO. APL: le crible d'Erathosthène.

35 FORTH: R4 R8; un métacompilateur simple; assembleur 6809 2ème part.; fBASE II pour ORIC ATMOS; VolksFORTH83 pour ATARI. dBASEII: présentation. I.A: rédactor.

36 MSDOS: prise en compte du clavier dans un fichier batch. IA: introduction à l'IA. FORTH: FBASEII sous MSDOS; variables chaînes exécutables. APL: le PGCD.

37 IA: FORTHLOG II; Logique et systèmes experts. FORTH: Handler TU58; éditeur plein écran pour MSDOS; manip. de piles et passage de paramètres.

38 FORTH: TURBO-Forth un produit NOTBORLAND; chargement de fichiers texte; virgule flottante en MVP-Forth. I.A: la logique bien formalisée.

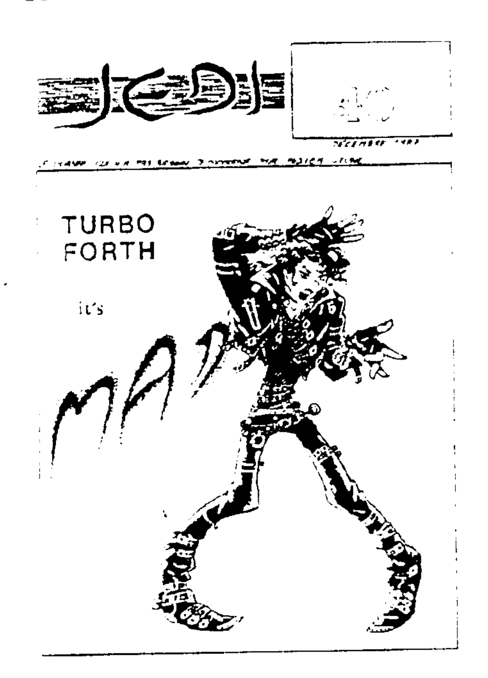
39 FORTH: Gestion de variables locales; fonctions graphiques CGA; gestion attributs minitel; libérateur de blocs écrans; décompression de fichiers F83; fonctions graphiques HERCULES; décomposition en facteurs premiers. LISP: récupération d'évaluation. PROLOG: logique binaire en Turbo-PROLOG. PASCAL: la pile heap de Turbo-Pascal.

40 FORTH: TURBO-Forth encore; accès aux commandes MSDOS en F83; fonction horloge et calendrier; div. et rac. carrées de grands nombres; traitement de chaînes. APL: tours de Hanoï. PASCAL: gestion de fichiers.

41 FORTH: Novix 4016; Commutation carte mono-cga; TURBO-Forth, suite; menus déroulants. LISP: robot logiciel. PROLOG: simulation de commandes de robot.

42 FORTH: du nouveau pour ATARI; Virgule flottante 83 Standard; Quadruple précision; modifications de Turbo-Forth. OCCAM: un langage pour les ordinateurs de 4ème génération. PROLOG: calcul de résistances.

TELEMATIQUE: JEDI SUR 3615; Connexion au réseau BTX.



43 FORTH: Variables locales; division çà précision infinie; Utilitaire de gestion de fichiers; Forth résident; Gestion carte HERCULES; Programmer sous GEM en VF83; Division et racine carrée en LM. COMMUNICATION: Liaison MINITEL-PC.

44 FORTH: virgule flottante; multi-fenêtrage; bibliothège GEM Volks-Forth; simulateur CPU 1802 RCA. PROLOG: carnet d'adresses à code. VO: Die Fileselector-box unter Volks-Forth; Classes in Forth.

45 FORTH: triturations binaires; traitement fichiers dBASE III; compilateur F79-F83; programmation lang. orienté objet; redirections E/S MS-DOS; gestion sons PC. TELEMATIQUE: accès à SAM•JEDI. VERSION ORIGINALE: Forth to the future.

46 FORTH: métacompilateur PHENIX TF83; scrutateur récursif de primitives; projet F32 (46); structures de données par enregistrement; fichiers binaires F83. DBASEIII: menu déroulant. VERSION ORIGINALE: forgettable internal names.

47 FORTH: gestion fenêtres par pile; téléchargement sur ATARI; conversion blocs>ASCII et invers. sur ATARI; virgule flottante sur ATARI; création accessoire sur ATARI; formatage disk sur ATARI. VERSION ORIGINALE: Use of a Forth Based Prolog (I)

48 FORTH: initiation à la gestion des fichiers séquentiels. VERSION ORIGINALE: Forth on the IBM, prime numbers; use of a Forth Based Prolog for Real Time Expert Systems (II)

49 FORTH: les suites de Syracuse; VERSION ORIGINALE: Compiling Prolog to Forth (III)

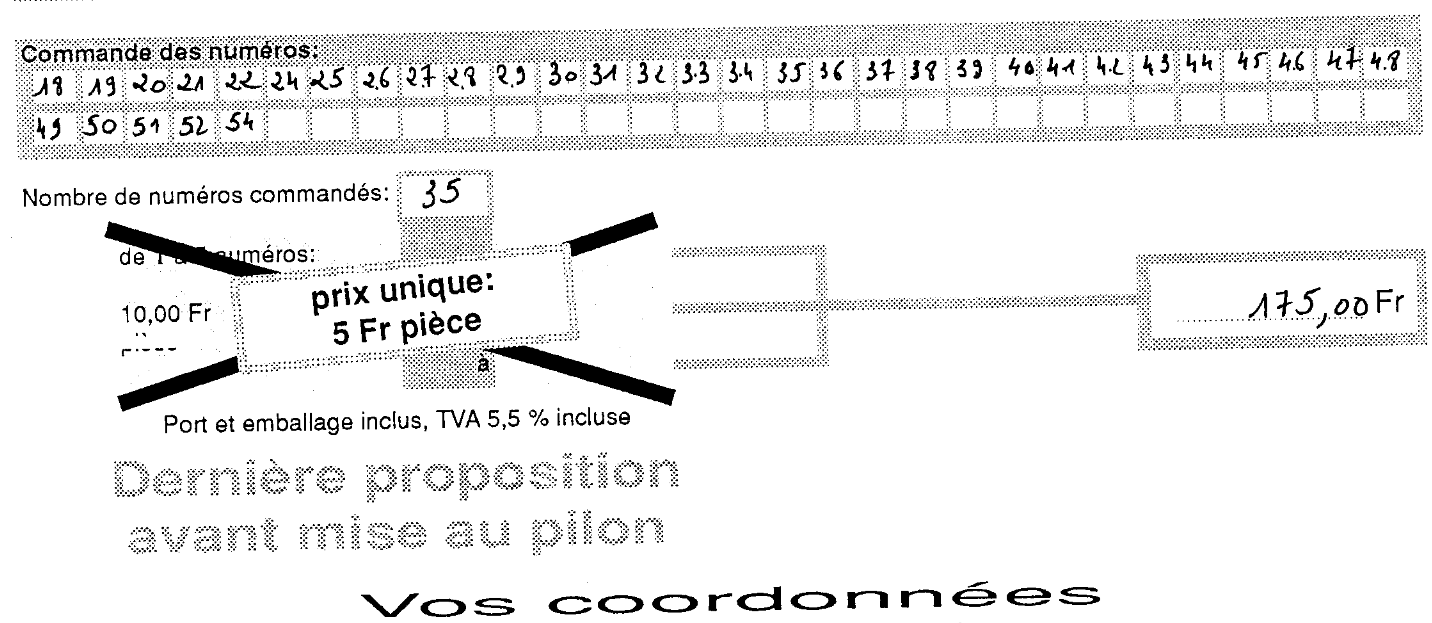
50 FORTH: PI jusqu'à 22000 décimales; éditeur de commandes DOS; utilitaire de menus déroulants; projet F32, spécifications. VERSION ORIGINALE: Compiling Prolog to FORTH (IV); Dragon, Fraktale für Forth.

51 FORTH: package virgule flottante, algorithme de Cordic pour TURBO-Forth.

52 FORTH: paramètres et pointeurs de pile; bannière écran; structure OUT-ENDOUT; visualisation d'images digitalisées; Turbo-Forth résident; tours de HANOI; Désassembleur 8086 pour TF83; création de listes de variables et de constantes.

53 animations 3D; accès aux interruptions; CORDIC, une bibliothèque de calcul en virgule flottante à précision variable.

54 (moins de 30 numéros disponibles) RTX 2000: le contrôle temps-réel; caractéristiques et architectures temps-réel; l'environnement de développement de la famille RTX. RESEAUX: réseaux et codes barres, l'exemple Cross-Bar



Commande des numéros:
18 19 20 21 22 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48
49 50 51 52 54

Nombre de numéros commandés: 35

de [redacted] numéros:
10,00 Fr

prix unique: 5 Fr pièce

175,00 Fr

Port et emballage inclus, TVA 5,5 % incluse

Dernière proposition avant mise au pilon

Vos coordonnées

Nom: RIBREUX Prénom: STEPHANE
Société:
Fonction:
Adresse: 1 RUE DENIS CORDONNIER
Code postal 62570 Commune: WIZERNES
tel: 21. 38. 46. 26 Fax:

Date 04/10/92 Signature

chèque à l'ordre de **REM CORP**, commande à expédier à **REM CORP, 17, rue de la Lancette 75012 PARIS**

EDITORIAL

Encore plus de PC semble être la devise maîtresse du dernier SICOB qui a eu lieu au CNIT DEFENSE. Parmis les exposants, le matériel IBM et compatible dominait largement.

Première constatation, les professionnels ne considèrent plus les micro-ordinateurs comme des gadgets, mais comme de véritables outils de travail. Pour preuve, la variété des applications présentées et leur destination était essentiellement à vocation professionnelle.

Seconde constatation, le recentrage du matériel. Dans la gamme IBM et compatibles, on ne présente plus que du matériel qui ne soit au moins du niveau des AT 286 et disposant d'au moins 40 Mo de capacité sur disque dur. Ceci est également vrai pour les portables.

Dernière constatation, la sensibilisation aux problèmes de sécurité des données. Tout d'abord, les virus et autres bombes logicielles sont de moins en moins tolérés. Ensuite, l'archivage, la conservation et la transmission des données sont sécurisées; le CD-ROM montre son nez, le streamer devient la norme et les protocoles avec identification du destinataire sont monnaie courante.

En dehors des systèmes IBM et compatibles, point de salut. Même si certaines machines, telles l'AMIGA 2000 ou le Mac INTOSH SE, sont intéressantes du point de vue performances, ce que je suis le premier à reconnaître, on ne peut plus espérer l'apparition d'un nouveau système qui deviendra la coqueluche des programeurs comme le furent en leur temps les APPLE II, ZX 81 et ORIC.

Autre phénomène, les virus. Que n'a-t-on pas proféré comme bêtises à ce sujet! Oui, un virus peut faire beaucoup de tort. Mais si comme moi vous avez été victime d'un atterissage de tête sur disque dur il y a quelque mois, sans être à l'origine d'une contamination virale binaire, vous n'auriez pas apprécié que certains logiciels, dBASE III+ pour le citer exprès, ne fonctionne plus après un RESTORE depuis les disquettes de sauvegarde périodique dont j'ai eu la bonne idée d'exécuter régulièrement. Certes, si LA COMMANDE ELECTRONIQUE vous remplace une disquette defectueuse parce que protégée, il risque de ne pas en être de même avec un produit logiciel également protégé, mais dont le constructeur a cessé ses activités.

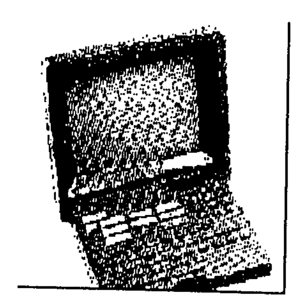
Je considère pour ma part que le plombage des logiciels est une atteinte à la liberté du client de disposer d'un bien qui lui appartient. Achèteriez-vous une voiture dont le capot permettant l'accès au moteur soit soudé sous prétexte que vous ne devez y intervenir en aucun cas? Le client doit être responsabilisé. S'il reproduit son logiciel illégalement, c'est à l'éditeur de contrôler et sanctionner les abus d'utilisation.

C'est en proposant des remises à jour à tarif préférentiel et en ne plombant pas les logiciels que les les éditeurs s'attacheront une clientèle fidèle et satisfaite, politique bien comprise par certains.

SOMMAIRE

Nos coordonnées: ASSOCIATION JEDI
17, rue de la Lancette
75012 PARIS
tél président: (33) 1-43.40.96.53
tél secrétaire: (33) 1-49.85.63.67

Télétel:
3615 SAM*JEDI (France, DOM, TOM)
(33) 36.43.15.15 (1200E71) (Etranger)



FORTH

ANIMATIONS 3 D

par Paul KOPFF

Adaptabilité: très difficile sur système non PC, du fait des accès au mode graphique EGA spécifique PC.
Disponibilité: téléchargement 3615 SAM*JEDI ou prochainement dans le module M9 TURBO-Forth

Chers Président et Secrétaire,

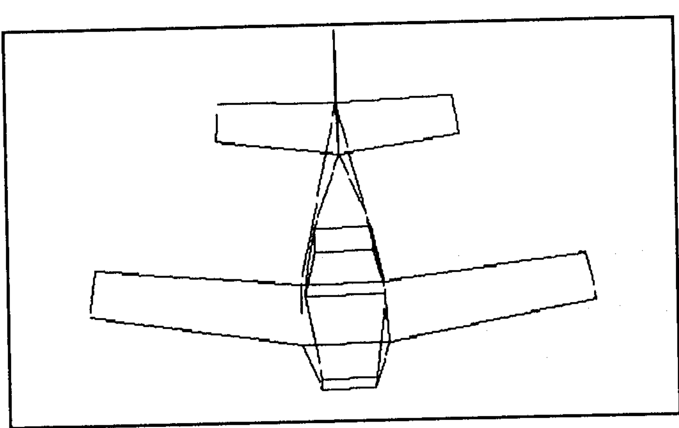
Votre éditorial en forme de coup de gueule contre l'informatique alibi à l'E.D.F. m'a fait beaucoup rire, car je fais partie de cette respectable institution; mais je plaide non-coupable, car mon job est le calcul numérique plutôt que l'informatique de gestion.

Jusqu'à présent, je faisais du FORTH chez moi, pour mon plaisir, sur des bécanes assez bizzares (ZX81 ou Sinclair QL), mais la pratique de ce langage finit par rendre assez intolérant à l'égard des autres (en particulier FORTRAN et BASIC, qui polluent encore très généralement les équipes de programmeurs "professionnels").

C'est pourquoi, j'essaye d'introduire FORTH comme langage pour certaines applications "sérieuses" dans mon Service (mais il faut user sa salive !).

J'ai installé TurboFORTH sur mon tout nouveau Toshiba T5200 (386/20Mhz... voyez qu'on a même les moyens de se payer autre chose que des timbres), et je dois avouer qu'il n'y a pas de plus agréable façon de partir à la découverte, dans les entrailles du petit monstre.

Je dis découverte, forcément, parce que pour le moment, pas moyen de se procurer la doc technique spécifique de cette bécane, et de plus, je suis pratiquement puceau dans le domaine des "compatibles".



Voici, non sans fierté, ma première petite application de TurboFORTH, que j'ai écrite en guise de démonstration pour faire voir à mes collègues le style et les performances de FORTH.

Dans mon Labo, on s'occupe d'Acoustique et de Vibrations de machines, et pour bien visualiser les phénomènes, on utilise souvent des représentations simplifiées d'objets tridimensionnels. J'ai donc écrit en TurboFORTH un ensemble de fonctions qui me permettent de décrire un objet "fil de fer", et de le représenter en perspective. Ce programme est incroyablement simple et compact, car il repose sur l'utilisation de fonctions vectorielles élémentaires (produits scalaires et vectoriels) et évite complètement le recours à la trigonométrie.

Cet ensemble de fonctions constitue le fichier 3DJ.FTH. Il doit includer un fichier contenant une fonction LINE comme par exemple GRAPHIC.FTH.

Bon, mais moi, je me suis fabriqué la mienne, dans le fichier LINA.FTH.

Pourquoi? J'ai testé la vitesse d'allumage d'un pixel par un appel à INT 10 du BIOS: 60 us, je me demande bien à quoi il passe tout ce temps? Du côté de chez APPLE, sur MAC II, la routine LINE met à peu près 15 us par pixel. Et LINE, en plus de l'allumage doit faire des calculs d'interpolation. Bon, sur ma bécane, en travaillant sans filet directement sur la mémoire écran, ma fonction LINE met 5.5 us par pixel.

La différence: soit un dessin comprenant environ 50 segments d'une longueur moyenne de 100 à 150 pixels, par segment je passe moins d'une ms à tracer, 2.5 ms à calculer la transformée perspective de ses extrémités (en FORTH); plus 1 ms pour les mots de haut niveau, résultat: je peux rafraîchir le dessin environ 5 fois par seconde. Avec LINE de GRAPHIC.FTH, je mets plus de 25 ms à tracer, le même temps pour le reste du boulot (négligeable), soit au total à peu près 1.3 sec par dessin.

J'ai donc pu écrire, par dessus les fonctions de 3DJ.FTH, une couche de fonctions d'animation par déplacements du point de vue; ceci se trouve dans le fichier 3DJA.FTH. Ca marche en mode EGA monochrome en basculant entre deux pages vidéo (en dessinant sur la page cachée). C'est très beau, sur une bécane rapide.

Améliorations possibles: codage en assembleur de la seule fonction VISION, la plus souvent appelée, devrait me permettre de gagner un petit facteur 2 sur la vitesse. Après, ça ne vaut sûrement plus le coup.

Amélioration de mes fonctions assembleur; ce sont d'infâmes bricolages, mais pourriez vous dans Jedi nous apprendre à nous servir de façon élégante et efficace de l'assembleur de TurboFORTH? Je me suis beaucoup inspiré des sources de KERNEL, pour le style, mais je ne pense pas encore avoir tout compris ou deviné, alors à l'aide S.V.P.!

Pour finir, le fichier DJ.FTH contient quelques fil_de_fer simples à titre d'exemples. Il include automatiquement 3DJ.FTH et 3DJA.FTH. le fichier 3DJ.FTH include en principe LINA.FTH., mais on peut le modifier pour utiliser GRAPHIC.FTH à la place (attention aux autres fonctions de LINA).

Pour tracer un fil de fer suivant le point de vue courant: le nommer simplement.

Pour changer de point de vue: donner 3 coordonnées position de l'observateur, et appeler POINT_DE_VUE.

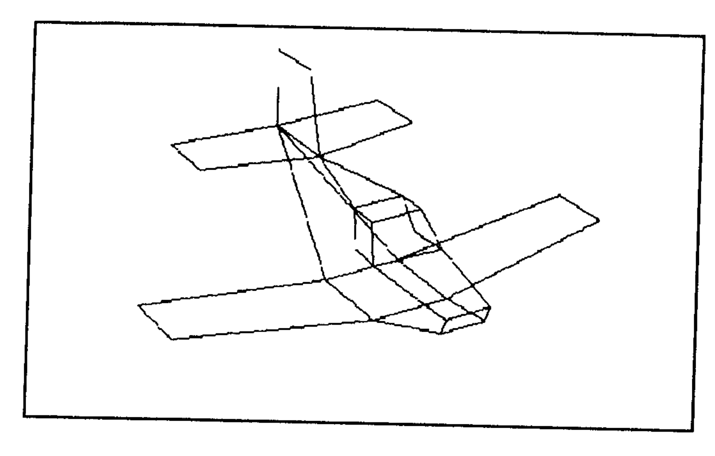
Pour faire bouger tout ça: nommer un fil de fer à la suite de REMUE, par exemple:

```
REMUE CUBE
```

les touches X Y Z servent à changer l'axe autour duquel tourne le point de vue et le sens de rotation. V fait tourner plus vite (+25%), D plus doucement (-25%). P rapproche l'observateur (-5%) et L l'éloigne (+5%). S stop ou step arrête le mouvement, et permet le pas à pas; A arrête le mouvement et passe en mode VGA, d'où une image fixe plus grande. Toute autre touche rend la main à FORTH.

Remarque: REMUE peut fonctionner avec autre chose qu'un FIL_DE_FER:

```
: TOUT cube octa rhombo ;
```



remue TOUT (TOUT n'est pas un fil_de_fer, mais une séquence)

Ce qui a principalement impressionné mes collègues dans cette application c'est la compacité du code (2714 octets entre HE et REMUE) et sa vitesse d'exécution. Je pense aussi les avoir convaincu qu'en FORTH on peut tout se permettre, et que la limite est l'imagination du programmeur. Créer des objets-fichiers-commandes (on ne sait plus comment les appeler) qui quand on les invoque, se dessinent eux-mêmes dans une perspective choisie...

LISTING 1: LINA.FTH

```
\ Paul KOPFF
\ fonction LINE en Assembleur 8086
\ pour ecran EGA ou VGA 2 couleurs
hex

\ Vidage de la page 0
code cls0
   0000 # ax mov  A000 # cx mov
   cx es mov  0000 # di mov
   4000 # cx mov  00F2 c, 00AB c,
next  end-code

\ vidage de la page 1
code cls1
   0000 # ax mov  A800 # cx mov
   cx es mov  0000 # di mov
   4000 # cx mov  00F2 c, 00AB c,
next  end-code

\ Allumage d'un pixel
\ En fonction de la page concernée, une instruction de cette
\ routine pointée par la variable PD sera modifiée par les
\ commandes L0 ou L1.

variable pd
code pset
            ax pop
            cx pop
         ax bx mov  \ dans BX, on transfère la   |2
            bx shl  \ coordonnée y, et on la     |2
            bx shl  \ multiplie par 5, puis on   |2
         ax bx add  \ ajoute A000, avant de      |3
      0F  # bh and  \ sauver cette valeur dans   |4
forth here 2+ pd ! assembler \ initialisation de PD
      A0  # bh  or  \ le registre segment ES     |4
         bx es mov  \ qui pointe donc la ligne   |2
         cx bx mov  \ dans BX, on transfère la   |2
            bx shr  \ coordonnée x, et on la     |2
            bx shr  \ divise par 8 pour repé-    |2
            bx shr  \ rer l'octet. Dans CL, on   |2
      07  # cl and  \ calcule le # du bit, que   |4
      80  # ax mov  \ l'on prend pour décaler    |4
         al cl ror  \ un bit dans AX.
                    \ Puis on              |8+4*#bit
   al 0 es: [bx] or \ affiche le pixel.          |16
next end-code

\ Tracé d'une ligne droite
\ En fonction de la page concernée, deux instructions
\ de cette routine, pointées par les variables PA et PB
\ seront modifiées par les commandes L0 ou L1.

variable pa
variable pb
code line
   dx pop  cx pop  bx pop
   bx -2 [rp] mov  \ y1
   1 # ax mov
   bx dx sub  0<
   if
      ax neg  dx neg
   then
   ax -4 [rp] mov  \ signe(y2-y1)
   bx pop
   bx -6 [rp] mov  \ x1
   1 # ax mov
   bx cx sub  0<
   if
      ax neg  cx neg
   then
   ax -8 [rp] mov  \ signe(x2-x1)
   0 # bx mov
   cx bx add
   dx bx add  0<>
   if  \ si (y2-y1)=(x2-x1)=0, c'est fini
      cx dx cmp  \ sinon:
      0<
      if  \ si (x2-x1)>(y2-y1)
         dx shl  dx -0C [rp] mov
         cx dx mov   cx di mov
         cx shl   cx -0A [rp] mov
         begin  \ boucle interpolation pixels
            -2 [rp] ax mov
            -6 [rp] cx mov
            -8 [rp] cx add
            -0C [rp] di sub  0<
            if
               -0A [rp] di add
               -4 [rp] ax add
            then
            ax -2 [rp] mov   cx -6 [rp] mov
            ax bx mov \ ds BX, on transfère la  |2
            bx shl  \ coordonnée y, et on la    |2
            bx shl  \ multiplie par 5, puis on  |2
            ax bx add \ ajte A000, avant de     |3
            0F # bh and
                    \ sauver cette valeur dans  |4
            forth here 2+ pa ! assembler
            A0 # bh or   \ le reg segment ES    |4
            bx es mov \ qui pointe donc la ligne|2
            cx bx mov \ dans BX, transfère la   |2
            bx shr  \ coordonnée x, et on la    |2
            bx shr  \ divise par 8 pour repé-   |2
            bx shr  \ rer l'octet. Dans CL, on  |2
   07 # cl and  \ calcule le # du bit, que  |4
   80 # ax mov  \ l'on prend pour décaler   |4
   al cl ror  \ un bit dans AX. Puis on  |8+4*#bit
   al 0 es: [bx] or  \ affiche le pixel.    |16
   dx dec  0=
         until
   else \ si au contraire (x2-x1)<(y2-y1)
   cx shl   cx -0C [rp] mov
   dx cx mov   cx di mov
   cx shl   cx -0A [rp] mov
   begin  \ boucle interpolation pixels
      -2 [rp] ax mov
    -6 [rp] cx mov
    -4 [rp] ax add
   -0C [rp] di sub
    0< if
   -0A [rp] di add
    -8 [rp] cx add
    then
   ax -2 [rp] mov
```

```
cx  -6 [rp] mov
      ax  bx  mov  \ dans BX, on transfère la  |2
          bx  shl  \ coordonnée y, et on la    |2
          bx  shl  \ multiplie par 5, puis on  |2
      ax  bx  add  \ ajoute A000, avant de     |3
   0F  #  bh  and  \ sauver cette valeur dans  |4
forth here 2+ pb ! assembler
   A0  #  bh   or  \ le registre segment ES    |4
      bx  es  mov  \ qui pointe donc la ligne  |2
      cx  bx  mov  \ dans BX, on transfère la  |2
          bx  shr  \ coordonnée x, et on la    |2
          bx  shr  \ divise par 8 pour repé-   |2
          bx  shr  \ rer l'octet. Dans CL, on  |2
   07  #  cl  and  \ calcule le # du bit, que  |4
   80  #  ax  mov  \ l'on prend pour décaler   |4
      al   cl   ror   \ un bit dans AX. Puis on
|8+4*#bit
  al  0  es:  [bx]  or  \ affiche le pixel.     |16
              dx  dec
    0= until
  then
then
next   end-code

\ Affichage de la page 0
code p0
  0500  #  ax  mov   10  int
next   end-code

\ Affichage de la page 1
code p1
  0501  #  ax  mov   10  int
next   end-code

\ adaptation des routines PSET et LINE
\ à la page 0 ou 1 (EGA).

: L0 A0 pa @ c! A0 pb @ c! A0 pd @ c! ;
: L1 A8 pa @ c! A8 pb @ c! A8 pd @ c! ;

decimal
\ initialisation des modes graphiques EGA
\ et VGA monochromes.

: vga 17 mode 320 HE ! 240 VE ! P0 L0 ;
: ega 16 mode 320 HE ! 175 VE ! ;

\ A ma connaissance, le mode VGA ne permet d'accèder qu'à la
\ page 0. Dommage !
```

```
LISTING 2:                 3DJ.FTH
\ Paul KOPFF

\ Perspective et visualisation de structures 3D
\ TurboFORTH

\ Cette application utilise une routine LINE
\ langage machine, qui trace 1 pixel / 5.5 us
\ au lieu de celle de GRAPHIC.fth (1 pixel / >60 us)

   variable HE  ( demi-largeur de l'écran en pixels )
   variable VE  ( demi-hauteur de l'écran en pixels )

   include lina.fth

\ Les fonctions PSET et LINE du fichier LINA.fth
\ n'utilisent pas INT 10 pour appeler la fonction PIXEL
\ du BIOS, mais modifient directement la mémoire d'écran
\ du Toshiba. Pour l'instant, ces fonctions supportent
\ les modes VGA et EGA monochrome du T5200 (mais je
\ pourrai les adapter à d'autres modes graphiques quand
\ j'aurai une documentation technique suffisante.)

\ Il n'est pas sûr que ces routines conviennent à
\ d'autres machines, mais j'ai préféré donner la priorité
\ à la vitesse. Rien n'empèche d'utiliser à la place,
\ LINE de GRAPHIC.fth, mais alors, on
\ voit les dessins se tracer; avec mes routines, c'est
\ instantané.

\ En tous cas, c'est encore à perfectionner!

\ Avant d'entrer dans le vif du sujet, une fonction
\ racine carrée: Son argument d'entrée est un entier
\ double (32 bits). Les résultats sont ( --- racine
\ carrée , résidu), où résidu est analogue au reste
\ de la division entière.

: sqrt ( d --- n,p | d = p*p + n )
  2dup or
  if -32768
    begin
      dup >r dup *d 2over d- r@ um/mod
      swap drop 2/ r> swap  ?dup
    while -
    repeat 1- >r r@ r@ *d d- drop r>
  then ;

\ Les approches classiques du problème de la projection
\ perspective reposent sur l'utilisation de matrices de
\ déplacements dans l'espace (translations et
\ rotations), qui ramènent la perspective au cas
\ particulier frontal (où le plan de l'image est
\ confondu avec un plan de coordonnées). La projection
\ perspective est alors très simplifiée; mais les
\ matrices de déplacements contiennent des cosinus
\ directeurs (et nécessitent donc des capacités de
\ calculs trigonométriques).

\ Au lieu d'ajouter à FORTH des fonctions
\ trigonométriques, j'ai adopté une autre approche
\ mieux adaptée aux possibilités intrinsèques de ce
\ langage.

\ En se référant à la figure 1:

\ Un observateur placé en P dirige sa vision vers O.
\ Tous les points M de la scène qu'il regarde seront
\ projetés par "sa" perspective sur un plan image PI
\ perpendiculaire à son axe de vision. Pour fixer ce
\ plan image, on supposera qu'il contient O et que sa
\ référence horizontale est son intersection avec le
\ plan horizontal absolu xOy de la scène.


\ Les points de la scène sont connus par leurs
\ coordonnées 3D, X, Y et Z, dans le repère de référence
\ absolu Oxyz. Leurs images dans la perspective de
\ l'observateur doivent être définies par leurs
\ coordonnées H et V, dans le repère Ohv du plan image.

\ Le problème de la transformation perspective revient
\ donc à cette transformation de coordonnées.

\ Elle peut s'exprimer par les opérations vectorielles
\ suivantes:

\             ->   ->      ->   ->        ->   ->
\             OI = OP ^ ( OP ^ MP ) / ( OP * MP )

\ où ^ désigne un produit vectoriel et * un produit
\ scalaire, dans l'espace des vecteurs 3D.

\ Cette formule peut être très facilement programmée,
\ car elle ne fait intervenir que les coordonnées de M
\ et P dans le repère absolu Oxyz. Les composantes du
\ vecteur OI qui en résulte sont également connues dans
\ ce même repère. Les exprimer dans le repère Ohv,
\ demande simplement deux produits scalaires de plus:

\                  ->   ->                 ->   ->
\              H = OI * OH             V = OI * OV

\ Le problème est donc résolu sans cosinus directeurs!
```

```
\ La programmation de cette application en FORTH comprendra
\ plusieurs couches logicielles:

\ 1ière couche: gestion et calculs pour vecteurs 3D

\ Définition d'un objet 3-vecteur
: 3-vecteur create 6 allot ;
    \ triplet de nombres de 16 bits

\ Manipulations entre pile et mémoire
: 3v! 6 0 do dup i + rot swap ! 2 +loop drop ;
: 3v@ 0 4 do dup i + @ swap -2 +loop drop ;

\ Manipulations sur la pile
: 3vdup  3 0 do 2 pick loop ;
: 3vover 3 0 do 5 pick loop ;
: 3vswap 3 0 do 5 roll loop ;
: 3vrot  3 0 do 8 roll loop ;
: 3vdrop 3 0 do   drop loop ;

\ Addition et soustraction de vecteurs 3D
: 3v+    3 roll + swap 3 roll + rot 3 roll + swap rot ;
: 3v-
   3vswap 3 roll - swap 3 roll - rot 3 roll - swap rot ;

\ Produits scalaires et vectoriels de vecteurs 3D

\ On apparentera ces fonctions aux opérateurs */ et al.
\ Données et résultats sur 16 bits, calculs
\ intermédiaires sur 32 bits.
\ V1,V2,s --- [V1*V2]/s scalaire ou [V1^V2]/s vecteur 3dim.

: 3v*
   >r 3 roll *d 2swap 4 roll *d
    rot 5 roll *d d+ d+ r> m/mod swap drop ;

: 3v^
    >r 4 pick over *d 3 pick 6 pick
    *d d- r@ m/mod swap drop
    4 roll 4 pick *d 3 roll 7 pick
    *d d- r@ m/mod swap drop
    5 roll 3 roll *d 4 roll 5 roll
    *d d- r> m/mod swap drop ;

\ module d'un vecteur 3D

: module
   dup *d rot dup *d d+ rot dup *d d+ sqrt swap drop ;

\ 2ième couche: fonctions de base de la
\ transformation perspective

\ La scène et l'observateur sont repérés dans Oxyz.
\ Un repère lié à la position de l'observateur est
\ construit:

3-vecteur P \ vecteur OP:
            \ point de vue de l'observateur dans Oxyz.

3-vecteur H \ horizontale OH:
            \ intersection du plan de projection
            \ perpendiculaire à OP avec le
            \ plan horizontal Oxy.

3-vecteur V \ frontale OV:
            \ perpendiculaire à OP et à OH.

variable  R \ module du vecteur OP.

variable  T \ module de la projection de OP sur Oxy.

variable  S \ facteur d'échelle S associé au rayon
            \ d'une sphère contenant la scène.

\ OP étant donné, le mot point_de_vue calcule OH et OV,
\ et des facteurs d'échelle R et T.
```

```
: point_de_vue ( OP --- )
   3vdup P 3v!     \ le vecteur OP est sauvé dans P
  3vdup module R ! \ son module est sauvé dans R
  3vdup 3vdup rot rot or \ vecteur OP est-il // à Oz ?
  if   \ sinon calcul de T, H et V
     2drop 0 module T !
     3vdup drop negate
     R @ T @ */ swap
     R @ T @ */ 0 H 3v!
     negate rot over
     T @ */ rot rot
     T @ */ T @ V 3v!
  else      \ si oui, trivial.
     drop   rot H 3v!
     negate rot rot V 3v!
  then
;
\ Le mot vision transforme les coordonnées 3D d'un
\ point M de la scène en les coordonnées 2dim de sa
\ projection perspective sur le plan OHV, en tenant \
compte du point de vue OP.

\ Un vecteur OM perpendiculaire à OP est transformé en
\ vraie grandeur.

: vision ( OM --- h,v )
  P 3v@   3vdup 3vrot 3v-    3vover 3vover
  R @ 3v* 3v^  P 3v@
  3vswap   R @ 3v^    3vdup    V 3v@ R @ 3v* >r
  H 3v@ R @ 3v* r> ;

\ Le mot ECRAN transforme ces coordonnées h,v brutes en
\ coordonnées pixel tenant compte du rayon de la scène
\ S et des résolutions
\ verticales et horizontales de l'écran (dans HE et VE).

: ecran ( h,v --- H,V )
  VE @ dup rot S @ */ - >r VE @ S @ */ HE @ + r> ;

\ 3ième couche: définition de fichiers fil_de_fer:
\ Un nombre CRD de sommets définis par leurs coordonnées
\ dans Oxyz,
\ Un nombre CNX de connections à tracer entre les
\ sommets, défini par la suite des numéros de sommets à
\ joindre, plume baissée si positif et plume levée si
\ négatif.

\ Les mots qui suivent définissent des objets de type
\ fil_de_fer. Quand on les crée, on y mémorise des
\ coordonnées et des connections. Quand on les invoque
\ (comme une commande FORTH), ils se dessinent
\ suivant le point de vue courant initialisé par
\ POINT_DE_VUE.

variable CRD    \ compteur pour le nombre de sommets
variable CNX    \ compteur pour le nombre de connections
: sommet ( #sommet --- H,V )
   6 * CRD @ + 3v@
   \ dans la table des sommets récup x,y,z
   vision
   \ transformation perspective x,y,z -> h,v
   ecran  \ mise à l'échelle écran h,v -> H,V ;
```

```
: segment ( Hi-1,Vi-1,adri --- Hi,Vi )
    @                   \ adri -> + ou - un numéro de sommet
    dup abs sommet      \ calcul de Hi,Vi
    rot 0>              \ était-ce + ou - ?
    if                  \ si + on trace la ligne
       2dup 2rot line
    else                \ si - on y va plume levée
       2swap 2drop
    then  ;

: dessin ( adr --- )
    dup CRD !
       \ dans CRD, début de la table des coordonnées
    dup 4 + @ S !     \ dans S, le rayon de la scène
       \ puis bouclage sur la table des connections:
    dup @ 1+ 6 * over +  dup rot 2+ @ 2* + swap 0 0 2swap
    do i segment 2 +loop
    2drop ;

\ fil_de_fer sera un nouveau nom de définition à utiliser en
\ conjonction avec C pour entrer des coordonnées X pour
\ entrer des connections et end_fil pour fermer le fichier.

: fil_de_fer ( s --- )
  create 4 allot  \ on réserve nombre de sommets     adr
  \ nombre de connections adr+2
    , ( virgule ! ) \ on sauve rayon de la scène   adr+4
   0 CRD ! 0 CNX !   \ on initialise les deux compteurs
  \ La suite du travail est effectuée par les fonctions C X
  \ et END_FIL
  \ A l'exécution, le dessin est automatiquement tracé
     does>    dessin  ;

\ Acquisition d'un sommet (3 coordonnées)
: C ( x,y,z --- )
   , , , ( 3 virgules ! )
   1 CRD +!     \ et incrémentation de CRD
;

\ acquisition d'une connection (+ ou - numéro de sommet)
: X ( +-#sommet --- )
   dup abs CRD @ <=      \ vérif si dans intervalle 1...CRD
   if , ( virgule ! )    \ si oui, acquisition
      1 CNX +!           \ et incrémentation de CNX
   then  ;

\ Pour clore un fichier de type fil_de_fer
: end_fil ( --- )
     last @ name> >body   \ on recherche le début du mot
     CRD @ over !         \ on sauve le nombre de sommets
     CNX @ swap 2+ !          \ on sauve le nombre de
connections ;

\ Le fichier source DEMO.fth contient un certain nombre
\ d'exemples de mots de type fil_de_fer.

\ Leur utilisation:
\ on définit un point de vue
2400 2200 3300 point_de_vue
\ et on les invoque simplement par leur nom.

\ REMARQUE: Tous les calculs opèrent sur des entiers 16
\ bits, avec parfois des phases intermédiaires sur 32 bits.
\ On a particulièrement fait attention à ne pas dégrader la
\ précision. Cependant, il est important que la précision
\ initiale soit correcte: Le vecteur POINT_DE_VUE devrait
\ avoir un module de 5000 à 25000, et la scène doit bien
\ remplir une sphère de rayon minimum 2500.
```

LISTING 3:   3DJA.FTH

```
\ Paul KOPFF

\ Animation de perspectives en temps réel TurboFORTH

\ Cette application constitue une couche logicielle
\ supplémentaire à 3DJ. Elle permet à un opérateur de
\ visualiser une structure fil_de_fer en changeant de
\ façon progressive et interactive de point_de_vue.
\ Les changements de point de vue qui seront considérés
\ - Rotation autour d'un des axes de coordonnées de Oxyz
\   Attention: c'est le point de vue qui tourne, pas
\   l'objet !
\ - Rapprochement ou éloignement suivant une direction

\ Ces fonctions sont surtout intéressantes quand on
\ dispose d'une machine très rapide: sur mon
\ T5200/20Mhz, des dessins simples sont animés de façon
\ pratiquement continue, et des fil_de_fers comprenant
\ une cinquantaine de segments de 100 à 150 pixels en
\ moyenne, sont rafraîchis à raison de 5 images par
\ seconde.

\ Etant donné une vitesse de rotation:
variable omega

\ On initialise un vecteur O dont le module est OMEGA et
\ qui est dirigé suivant Ox, Oy ou Oz.
3-vecteur O
: rx ( omega --- )    0 0              0 3v! ;
: ry ( omega --- )    0 0 rot rot      0 3v! ;
: rz ( omega --- )    0 0 rot          0 3v! ;

\ La rotation d'un vecteur P autour de l'axe portant le
\ vecteur O s'exprime par une séquence de calculs
\ comprenant en particulier des produits vectoriels de
\ P et O.
: pas
    P 3v@ O 3v@    3vover 3vover
    R @ 3v^    R @ 3v^    2/ rot 2/ rot 2/ rot   3v-
    3vdup O 3v@ R @ 3v^    3v+    point_de_vue ;

\ Association des touches X Y Z aux changement d'axes
\ de rotation et aux changement de sens de rotation.
variable axe
3 case: taxe rx ry rz ;
: raxe omega @ axe @ taxe ;
: tx omega @ negate omega ! 0 axe ! raxe ;
: ty omega @ negate omega ! 1 axe ! raxe ;
: tz omega @ negate omega ! 2 axe ! raxe ;

\ Association des touches V et D aux variations de
\ vitesse de rotation
\   V plus vite
\   D plus doucement

5 4 2constant po
: tv omega @ dup abs 500 <
  if po      */ then omega ! raxe ;
: td omega @ dup abs    5 >
  if po swap */ then omega ! raxe ;

\ Association touches P et L variations de perspective
\   L plus loin
\   P plus près
21 20 2constant pp
: tl R @ 25000    <
   if P 3v@ 3 0
      do pp  */ rot loop point_de_vue
   then ;
: tp R @ S @ 2*   >
   if P 3v@ 3 0
      do pp swap */ rot loop point_de_vue
   then ;
variable objet

\ Deux autres touches fonctions
\   S stop ou step: arrêt sur image et pas à pas
\   A aggrandissement d'une image fixe EGA -> VGA
: ts begin key? until ;
: ta vga objet @ execute ts ega ;

\ Toute autre touche rend la main à FORTH
```

```
\ Fonction de selection des touches fonction
: modif
          key? if
           key case ascii x of tx endof
                    ascii y of ty endof
                    ascii z of tz endof
                    ascii v of tv endof
                    ascii d of td endof
                    ascii l of tl endof
                    ascii p of tp endof
                    ascii s of ts endof
                    ascii a of ta endof abort endcase
               then ;

\ Interface utilisateur

\ Le mot REMUE attend comme argument un mot de type
\ fil_de_fer

\              par exemple REMUE CUBE

\ Il sauve son adresse d'exécution dans la variable OBJET
\ puis boucle sur son exécution alternée dans les 2 pages
\ EGA tout en détectant les manoeuvres des touches fonction

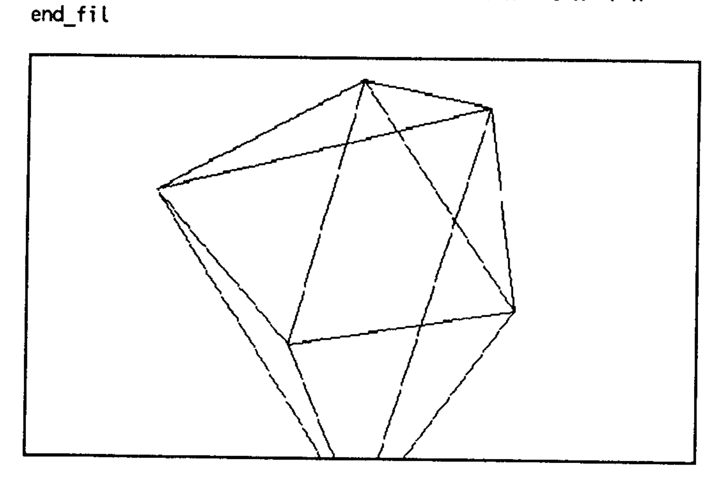
: remue   ' objet ! ega
   begin
       P0 CLS1 L1 objet @ execute pas modif
       P1 CLS0 L0 objet @ execute pas modif
   again ;
```

FICHIER 4: DJ.FTH

```
echo off
include 3DJ
include 3DJA

\ dodecaedre regulier

800 fil_de_fer dodeca
-459 0 601 c -142 -436 601 c 371 -270 601 c 371 270 601 c
-142 436 601 c -742 0 142 c -601 -436 -142 c -229 -706 142
c
229 -706 -142 c 601 -436 142 c 742 0 -142 c 601 436  142 c
229 706 -142 c -229 706 142 c -601 436 -142 c -371 270 -601
c
-371 -270 -601 c 142 -436 -601 c 459 0 -601 c 142 436 -601
c
-18 x  17 x  7 x  6 x  1 x  5 x  -6 x  15 x -17 x  16 x
 15 x  14 x  5 x  4 x -14 x  13 x -16 x  20 x  13 x  12 x
  4 x  3 x -12 x  11 x -20 x  19 x  11 x  10 x  3 x  2 x
-10 x  9 x -19 x  18 x  9 x  8 x  2 x  1 x  -8 x  7 x
end_fil
```



```
\ rhomboèdre
2100 fil_de_fer rhombo
1000 -1000  0 c 1000 0 1000 c 1000 1000 0 c
1000 0 -1000 c 0 -1000 -1000 c 0 -1000  1000 c
0 1000 1000 c 0 1000 -1000 c -1000 0 -1000 c
-1000 -1000 0 c -1000 0 1000 c -1000 1000 0 c
-1 x 2 x 3 x 7 x 12 x 11 x 7 x 2 x 6 x 11 x 10 x 6 x 1 x
5 x 10 x 9 x 12 x 8 x 3 x 4 x 8 x 9 x 5 x 4 x 1 x
end_fil

\ cube
2100 fil_de_fer cube
1000 1000 1000 c 1000 1000 -1000 c 1000 -1000 1000 c
1000 -1000 -1000 c -1000 1000 1000 c -1000 1000 -1000 c
-1000 -1000 1000 c -1000 -1000 -1000 c
-1 x 5 x 6 x 8 x -5 x 7 x 8 x 4 x -7 x 3 x 4 x 2 x -3 x
1 x 2 x 6 x
end_fil

\ octaedre regulier

2100 fil_de_fer octa
2000 0 c 0 2000 0 c 0 0 2000 c
-2000 0 0 c 0 -2000 0 c 0 0 -2000 c
-3 x 1 x 2 x 6 x 5 x 4 x 3 x 2 x 4 x 6 x 1 x 5 x 3 x
end_fil

\ avion
: xa 16 2 do i x loop ;
3000 fil_de_fer avion
 2600 -350 300 c 1300 -600 0 c 1000 -3500 300 c 0 -3500
300 c
 0 -600 0 c -2600 0 1000 c -2600 -1800 1050 c -1600
-1800 1050 c
-1300 0 1000 c 400 -400 1000 c 900 -400 1000 c 1200 -550
600 c
 1200 550 600 c 900 400 1000 c 400 400 1000 c -1600 1800
1050 c
-2600 1800 1050 c -1600 0 2000 c -2600 0 2000 c 0 600 0
c
 0 3500 300 c 1000 3500 300 c 1300 600 0 c 2600 350 300
c
 2700 -350 500 c 2700 350 500 c 600 -500 600 c 600 500
600 c
-1 x xa 9 x 16 x 17 x 6 x 9 x 18 x 19 x 6 x 20 x 21 x 22
x 23 x 20 x 5 x 2 x 23
 x 24 x -25 x 26 x 24 x 1 x 25 x 12 x 27 x 10 x 15 x 28
x 13 x 26 x -11 x 14 x
end_fil

: reset 7000 7000 7000 point_de_vue 220 dup omega ! rx
vga ;

echo on reset remue avion
```

■

LIBRE PROPOS

ACCES AUX INTERRUPTIONS...

par Marc PETREMANN

Dans l'article ANIMATIONS 3 D, Paul KOPFF touche du doigt un problème crucial: doit-on utiliser systématiquement les fonctions d'interruptions ou travailler en accès mémoire lors d'opérations travaillant en mode graphique?

Sa solution est certes de loin la meilleure sur le plan du rendement. Mais alors, pourquoi dispose-t-on d'une routine d'interruption?

A l'origine, le premier système IBM PC (les clones n'existant même pas) ne disposaient d'aucun mode graphique ni gestion de couleur. Mais malins comme le sont les concepteurs de BIG BLUE, ils ont anticipé en mettant à disposition une case avec un code d'accès (INT 10, fonction 00H pour le mode vidéo, 05H pour sélection de page, OCH pour traçage d'un point). Ce principe garantit à tout concepteur que sa fonction graphique tournera avec n'importe quelle carte vidéo existante ou à venir.

Comment fonctionne une interruption? Une interruption est appelée dès que le code opérateur INT est activé et passe comme paramètre le numéro de l'interruption choisie. Il existe 256 interruptions possibles. Dès lors, le processeur 80xx ou 80xxx va lire dans le premier segment mémoire de la RAM une suite de quatre octets situés à l'offset 'numéroINTx4'. Ainsi pour l'interruption 10h (16d), l'adresse du vecteur est située en 0000:0040h; le contenu de ces quatres octets dépend ensuite du BIOS de votre système. Un programme peut modifier ce contenu pour rediriger une interruption.

Le constructeur déconseille de passer outre une fonction d'interruption; si c'est le cas, il ne garantit plus la portabilité de vos programmes.

Mais lisez plutôt l'anecdote suivante:

Une revue mensuelle m'a commandé très récemment deux articles concernant le test de systèmes portables: un SHARP PC 286, 40 Mo DD, écran LCD émulation VGA et un portable IBM PS/2 80386 MO Mhz, 140 Mo DD, écran à plasma.

Pendant que je bricolais un peu sur le SHARP, un quidam passant par la pièce où j'opérais, a lancé un programme de test standard (créé par PC Magazine US) qui vous vérifie la vitesse d'affichage, l'indice NORTON, et tout le bazar et recrache un topo complet prêt à recopier, le tout en deux minutes; rien d'anormal.

Moi, bête programmeur FORTH que je suis, et ne disposant pas de ces merveilleux logiciels d'exploration, j'installe TURBO d'abord sur le SHARP. Tiens, si j'essayais le mode graphique VGA, si bien émulé. Pas de problème jusqu'au mode 18 (VGA 640x400 16 couleurs). OUUUIIILLLEEEE!!! Le curseur ne se met pas tout en haut à gauche à l'exécution d'une simple commande 0 0 PSET mais au milieu dernier quart droite de l'écran!!!! Aurai-je mal tapé quelque chose? Mon programme est-il-bogué? Aussitôt j'avise le COMPAQ 80286 placé derrière moi. Je relance TURBO et GRAPHIC et teste 0 0 PSET: le pixel s'allume en haut à gauche, ce qui est le comportement normal et attendu de ma routine graphique.

Une fois mon papier pondu au sujet du SHARP, je repasse le lendemain pour décortiquer le PS/2 d'IBM. Installation de TURBO-Forth et premiers essais: une boucle 0 à 1 millions en 0,69 sec... pas mal! Voyons maintenant le mode graphique, également émulé pour ce qui est du VGA:

```
INCLUDE GRAPHIC
18 MODE
15 GCOLOR
0 0 PSET
```

Je ne refais pas le OUUUIIILLLEEEE!!! mais plutôt un M.... bien senti! J'appelle le service technique IBM:" vous n'avez rien vu d'anormal en mode VGA 18?" (je résume, car les grosses sociétés, c'est toujours 10 mn de standard et secrétaires avant d'avoir un technicien compétent).

Pour moi, l'affaire est grave. Plusieurs réponses possibles:

- le driver d'émulation VGA du SHARP et du PS/2 IBM sont identiques (qui a copié sur qui?) et bogués,
- j'ai du oublier de remettre à zéro un paramètre; mais alors pourquoi celà fonctionne-t-il bien sur COMPAQ?

A ce jour, je n'ai pas la réponse, mais je relancerai les services "compétents" d'IBM. En conclusion: les tests standards ne détectent pas les bogues. Grâce à TURBO et le hasard, j'ai mis le doigt sur un fait quasi-établi chez les constructeurs de machines et chez le pape IBM lui-même: on définit une routine d'interruption, mais on ne l'utilise pas!! Qui plus est, cette routine est fausse, mais on s'en moque...

Et pourquoi les constructeurs s'en priveraient-ils? Tous les programmes doivent court-circuiter les interruptions pour être performants.

Alors pourquoi encombre-t-on les machines avec des BIOS monstrueux et des DOS mamouthesques puisque les logiciels apportent leurs propres routines.

Pour mémoire, avec un CP/M tenant en 12K, on arrivait déjà à faire des merveilles. Puis avec DOS 3.x on franchit les 64k de mémoire occupée.. occupée à quoi?

Et maintenant avec OS/2 prenant pas loin de 500 Ko, nous ne sommes pas loin d'occuper un espace mémoire aussi vaste que celui utilisé par un compilateur ADA!!!

Je crie: HALTE A L'ESCROQUERIE!! Arrêtez de pondre des machines ressemblant à un VELOSOLEX motorisé avec un V12 de CHEVROLET. Ceci ressemble à la course aux armements.

Conclusion: je ne conseille pas l'achat de tel ou tel système, ni ne souhaite casser la réputation d'un quelconque constructeur. Je constate, plus que je ne dénonce, les aberrations auxquelles nous nous trouvons de plus en plus fréquemment confrontés en tant que programmeurs. ■

TELEMATIQUE

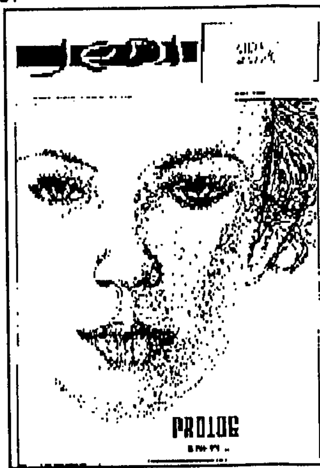
CONTENU DU FORUM SAM*JEDI

FORTH7 02.08.89 21h42
RENTREE EN FORTH DANS FD DE JUILLET:

SILICON COMPOSERS lance le SC32 premier Stack-Chip muP Forth 32-bits:
- un cycle par instruction a 8 ou 10 MHz
- 16 gigabyte non-segmented data space
- 2 gigabyte non-segmented code space
- piles illimitees (sauf taille memoire)
- Non-Multiplexed 32-bit address bus and data bus
- $195 le chip 84-pin ceramic CMOS
- $1995 la carte coprocesseur // pour PC/XT/AT/386 avec 64Ko zero wait RAM livree avec SC/Forth32 extension a 32 bits du F83 Standard.

SECRETAIRE 05.08.89 23h37
SORTIE DU MODULE 7 (APRES LE MODULE 8...):
- DESASSEMBLEUR TF83
- TOURS DE HANOI
- POURSUITE (UN JEU STYLE PACMAN...)
- VERSION AMELIOREE DE HALT.FTH
- VISU D'IMAGES DIGITALISEES

ET QUELQUES AUTRES FICHIERS. EN TOUT, PRES DE 100 KO DE SOURCES EN FORTH. PRIX 37 FR EN TIMBRES DE PREFERENCE A COMMADER A ASSOCIATION JEDI 17, RUE DE LA LANCETTE 75012 PARIS.



SECRETAIRE 05.08.89 23h41
CHALLENGE: JE DONNE TOUJOURS DES TRUCS MAIS MAINTENANT JE VOUS FAIT BOSSER:

1 MODULE 7 EN CADEAU AU PREMIER QUI ME DONNE UNE DEFINITION DE -PICK SUR CE FORUM (J'EXCLUE DU CHALLENGE

(suite page 17)

CORDICFP : UNE BIBLIOTHEQUE DE CALCUL EN VIRGULE FLOTTANTE A PRECISION PARAMETRABLE

par CHRISTOPHE LAVARENNE
ingénieur conseil en informatique temps réel

EN BREF

Une bibliothèque originale de calcul en virgule flottante est présentée. L'algorithme unifié CORDIC, sur lequel elle est basée, est d'abord présenté et justifié, montrant sa simplicité, son auto suffisance quel que soit le nombre de bits significatifs requis, et son efficacité pour la manipulation des nombres complexes. Les choix faits pour son implantation sont ensuite discutés, au niveau de la portabilité sous Forth83, de l'éfficacité, et de la compatibilité avec les standards IEEE. Enfin, quelques mesures de performances sont comparées avec celles d'une bibliothèque de calcul en virgule flottante classique basée sur les approximations polynomiales.

INTRODUCTION

CORDICFP est basée sur l'algorithme unifié de Volder, qui traite la plupart des fonctions mathématiques élémentaires dans un cadre unique basé sur les "rotations généralisées" (CORDIC est un acronyme pour COordinate Rotation DIgital Computing).
L'algorithme est facile à implanter, en logiciel comme en matériel, pour un nombre quelconque de bits de précision, car les seules opérations requises sont les décalages (bit-shifts), accumulations (addition et soustraction), et rappel de constantes, elles-mêmes précalculées au moyen de ces seules opérations.
La complexité spatio-temporelle de l'algorithme est pratiquement proportionnelle au carré du nombre de bits de précision (comme pour une simple multiplication), ce qui le rend plus efficace que les méthodes d'approximations polynomiales pour les fonctions transcendentales.
Comme le domaine de convergence de l'algorithme est limité, des changements d'échelle pré- et post-opératoires sont nécessaires pour les calculs en virgule flottante. La représentation la plus commune des nombres réels en virgule flottante, avec mantisse normalisée et exposant binaire séparés, est la mieux adaptée à l'algorithme et aux changements d'échelle.

BASES MATHEMATIQUES

Les fonctions hyperboliques et trigonométriques sont liées par les relations :

$$\sinh(z) = -j.\sin(j.z)$$
$$\cosh(z) = \cos(j.z) \qquad \text{avec: } j^2 = -1$$
$$\text{atanh}(z) = -j.\text{atan}(j.z)$$

Ces fonctions élémentaires peuvent être présentées dans un cadre unique, en introduisant les fonctions suivantes paramètrées par m, avec m=1 dans le cas trigonométrique, m=-1 dans le cas hyperbolique, et également définies pour m=0, cas dit linéaire :

$$\text{Gsin}(z) = m^{-1/2}.\sin(m^{1/2}.z) \qquad \text{Gsin}(z) \xrightarrow[m\to 0]{} z$$

$$\text{Gcos}(z) = \cos(m^{1/2}.z) \qquad \text{Gcos}(z) \xrightarrow[m\to 0]{} 1$$

$$\text{Gatan}(z) = m^{-1/2}.\text{atan}(m^{1/2}.z) \qquad \text{Gatan}(z) \xrightarrow[m\to 0]{} z$$

$$\text{Gcos}^2(z) + m.\text{Gsin}^2(z) = 1$$

$$\text{Gtan}(z) = \text{Gsin}(z) / \text{Gcos}(z)$$

Christophe LAVARENNE, 12 rue du Docteur Vuillième, 92190 MEUDON, FRANCE. Tél: (33-1) 45071341

Rotations généralisées

Les rotations généralisées sont définies dans le cadre d'un ensemble de systèmes de coordonnées paramètrés par m, dans lesquels le rayon R et l'angle A du vecteur P=(x,y) sont définis par :

$R = \text{signe}(x).(x^2+m.y^2)^{1/2}$ => $x = R.\text{Gcos}(A)$ signe(x>=0)=1
$A = \text{Gatan}(y/x)$ $y = R.\text{Gsin}(A)$ signe(x<0)=-1

m=1 (trigonométrique)	m=0 (linéaire)	m=-1 (hyperbolique)
$R = \text{signe}(x).(x^2+y^2)^{1/2}$	$R = x$	$R = \text{signe}(x).(x^2-y^2)^{1/2}$
$A = \text{atan}(y/x)$	$A = y/x$	$A = \text{atanh}(y/x)$
$x = R.\cos(A)$	$x = R$	$x = R.\cosh(A)$
$y = R.\sin(A)$	$y = R.A$	$y = R.\sinh(A)$

Toute rotation d'angle da et conservant le rayon, transformera le vecteur P en P' suivant :

```
A'= A + da      =>    | x'|     | Gcos(da)   -m.Gsin(da) |   | x |
R'= R                 | y'| =   | Gsin(da)      Gcos(da) | . | y |
```

que l'on peut aussi écrire :

```
| x'|                 |    1        -m.Gtan(da) |   | x |
| y'| = Gcos(da) .    | Gtan(da)         1      | . | y |
```

Les rotations élémentaires suivantes, semblables au précédentes bien que ne conservant pas le rayon pour m=1 et m=-1, sont beaucoup plus intéressantes parce qu'elles ne nécessitent que des décalages binaires et des accumulations (additions ou soustractions) :

```
| x"|     |   1     -m.s.2^-n |   | x |     <=>     A"= A + s.a
| y"| =   | s.2^-n      1     | . | y |             R"= k.R
```

$s = \pm 1$ (signe de chaque rotation élémentaire)
$a = \text{Gatan}(2^{-n})$ (module de chaque rotation élémentaire)
$k = 1/\text{Gcos}(a) = [1 + m.\text{Gtan}^2(a)]^{1/2} = (1 + m.2^{-2n})^{1/2}$ (facteur radial)

m=1 (trigonométrique)	m=0 (linéaire)	m=-1 (hyperbolique)
$a = \text{atan}(2^{-n})$	$a = 2^{-n}$	$a = \text{atanh}(2^{-n})$
$k = (1 + 2^{-2n})^{1/2}$	$k = 1$	$k = (1 - 2^{-2n})^{1/2}$

La composition de plusieurs rotations élémentaires produit une variation angulaire A égale à la somme arithmétique des a_i élémentaires, et un facteur radial K égal au produit des k_i élémentaires :

$$A = \sum_i s_i.a_i = \sum_i s_i.\text{Gatan}(2^{-n_i})$$
$$K = \pi_i k_i = \pi_i (1 + m.2^{-2n_i})^{1/2}$$

On montre par la suite qu'il existe pour chaque m une séquence déterminée de rotations élémentaires, c'est-à-dire une séquence déterminée de n_i, pour laquelle :

- il existe une séquence s_i pour tout A tel que $|A| < \sum_i a_i$
- $K = 2^m$, réduisant ainsi à un seul décalage la réduction du facteur radial

Ainsi, pour une telle séquence de N rotations élémentaires, une rotation quelconque se fera en seulement 2N décalages et 2N accumulations, et la variation angulaire correspondante sera l'accumulation de N constantes (les $\text{Gatan}(2^{-n_i})$).

Christophe LAVARENNE, 12 rue du Docteur Vuillième, 92190 MEUDON, FRANCE. Tél: (33-1) 45071341

Identités

Ces identités mathématiques permettront le calcul des constantes et les changements d'échelle permettant de ramener aisément les nombres codés en virgule flottante dans les domaines de convergence de l'algorithme :

avec $|z|\leq 1$:

$$Gatan(z) = \sum_{i\geq 0}(-m)^i.z^{2i+1}/(2i+1)$$

avec M dans [-1,-1/2[U[1/2,1[($|M|\leq 1$ converge):

$$(M_x.2^{E_x}).(M_z.2^{E_z}) = (M_x.M_z).2^{E_x+E_z}$$
$$(M_y.2^{E_y})/(M_x.2^{E_x}) = (M_y/M_x).2^{E_y-E_x}$$

avec $z = R + Q.(\pi/2)$ ($\pi/2=1.57$, $|R|\leq 1.74$ converge):

avec r<4 et Q=4.q+r	sin(z) =	cos(z) =
quand r = 0	sin(R)	cos(R)
quand r = 1	cos(R)	-sin(R)
quand r = 2	-sin(R)	-cos(R)
quand r = 3	-cos(R)	sin(R)

$$\tan(z) = \sin(z) / \cos(z)$$

quand $|z|>1$ (atan(z) converge pour $|z|\leq 1$):

$$atan(z) = (\pi/2) - atan(1/z)$$

avec $z = R + Q.\log_e(2)$ ($\log_e(2)=0.69$, $|R|\leq 1.13$ converge):

$$\sinh(z) = 2^{Q-1}\{[\cosh(R) + \sinh(R)] - 2^{-2Q}[\cosh(R) - \sinh(R)]\}$$
$$\cosh(z) = 2^{Q-1}\{[\cosh(R) + \sinh(R)] + 2^{-2Q}[\cosh(R) - \sinh(R)]\}$$
$$\exp(z) = e^z = 2^Q[\cosh(R) + \sinh(R)]$$

$$\tanh(z) = \sinh(z) / \cosh(z)$$

avec $z = M.2^E$, $1/2\leq M<1$ ($0.10\leq M\leq 9.58$ converge):

$$\ln(z) = \log_e(z) = -2.atanh[(1-M)/(1+M)] + E.\ln(2)$$

avec $z = M.2^E$, E pair, $1/2\leq M<2$ ($0.03\leq M\leq 2.42$ converge):

$$z^{1/2} = 2^{(E/2)-1}.[(1+M)^2-(1-M)^2]^{1/2}$$

avec $|z| = 1-M.2^{-E}$, $E\geq 0$ pair, $1/2\leq M<2$:

$$atanh(z) = atanh[(1-M+z)/(1+M+z)] + (E/2).\ln(2)$$
$$(1-z^2)^{1/2} = 2^{-(E/2)-1}.[(1+M+z)^2-(1-M+z)^2]^{1/2}$$

avec $|z|<1$ ($asin(1)=\pi/2$):

$$asin(z) = atan[z/(1-z^2)^{1/2}]$$
$$acos(z) = (\pi/2) - asin(z)$$

Christophe LAVARENNE, 12 rue du Docteur Vuillième, 92190 MEUDON, FRANCE. Tél: (33-1) 45071341

L'ALGORITHME CORDIC

La séquence s_i est déterminée par approximations successives :
chaque rotation élémentaire a_i=Gatan(2^{-n_i}) est accumulée à l'angle z (de la rotation à approximer) avec un signe s_i choisi de manière à forcer z vers zéro.

Rotation

Les approximations successives et les rotations élémentaires sont décrites par les équations itératives suivantes :

$$x_{i+1} = x_i - m.s_i.y_i.2^{-n_i}$$
$$y_{i+1} = y_i + s_i.x_i.2^{-n_i}$$
$$z_{i+1} = z_i - s_i.a_i$$

avec s_i tel que : $|z_{i+1}| = | |z_i| - a_i |$
c'est-à-dire : si $z_i<0$ alors $s_i=-1$ sinon $s_i=1$
ou encore : s_i = signe(z_i)

$P_0=(x_0,y_0)$ subit une rotation d'angle z_0, donnant $P_N=(x_N,y_N)$ avec :

$$x_N = K.[x_0.\mathrm{Gcos}(z_0) - m.y_0.\mathrm{Gsin}(z_0)]$$
$$y_N = K.[x_0.\mathrm{Gsin}(z_0) + y_0.\mathrm{Gcos}(z_0)]$$

m=0
$$x_N = x_0$$
$$y_N = y_0 + x_0.z_0$$

Vectorisation

Les mêmes équations itératives et séquences de rotations élémentaires peuvent être utilisées pour faire subir au vecteur P=(x,y) une rotation d'angle A=-Gatan(y/x), simplement en forçant A, c'est-à-dire y (au lieu de z) vers zéro :

$$x_{i+1} = x_i - m.s_i.y_i.2^{-n_i}$$
$$y_{i+1} = y_i + s_i.x_i.2^{-n_i}$$
$$z_{i+1} = z_i - s_i.a_i$$

avec s_i tel que : $|y_{i+1}| = | |y_i| - |x_i.2^{-n_i}| |$
c'est-à-dire : si $(y_i/x_i)\geq 0$ alors s_i =-1 sinon s_i =1
ou encore : s_i = -signe(y_i).signe(x_i)

$P_0=(x_0,y_0)$ est vectorisé (subit une rotation de -A), pour donner :

$$R = x_N = K.\mathrm{signe}(x_0).(x_0^2+m.y_0^2)^{1/2}$$
$$A = z_N - z_0 = \mathrm{Gatan}(y_0/x_0)$$

m=0
$$x_N = x_0$$
$$z_N = z_0 + (y_0/x_0)$$

Critère de convergence

L'algorithme sera dit converger en N itérations si :

(1): $|A_N| \leq a_{N-1}$ rotation: $A_N=z_N$ vectorisation: A_N=Gatan(y_N/x_N)

A chaque itération i, la somme des rotations élémentaires restantes doit être suffisante pour amener A_i à au plus a_{N-1} de zéro :

$$(2): |A_i| - \sum_{j=i}^{N-1} a_j \leq a_{N-1}$$

même dans le cas extrême où (A_i = 0) <=> ($|A_{i+1}| = a_i$), soit d'après (2) :

$$(3): a_i \leq a_{N-1} + \sum_{j=i+1}^{N-1} a_j$$

En posant ($-a_i < |A_i|-a_i$) et en combinant avec (2) et (3), on obtient :

$$(4): | |A_i| - a_i | \leq a_{N-1} + \sum_{j=i+1}^{N-1} a_j$$

Christophe LAVARENNE, 12 rue du Docteur Vuillième, 92190 MEUDON, FRANCE. Tél: (33-1) 45071341

Comme l'algorithme par approximations successives garantit :

$$(5): |A_{i+1}| = | |A_i| - a_i |$$

on en déduit que si (2) et (3) sont vérifiées pour i, alors (2) est également vérifiée pour i+1.

En conséquence, pour que (1) soit verifiée, il faut et il suffit :
- que (2) soit vérifiée pour i=0 (ce qui borne le domaine de convergence de l'algorithme)
- que (3) soit vérifiée pour tout i :

Pour m=0, la relation (3) est vérifiée avec la séquence ($n_{i+1} = n_i+1$)
car : $a_i = 2^{-n_i} = 2.2^{-(n_i+1)} = 2.a_{i+1} = a_{i+1}+2.a_{i+2} = \ldots = a_{i+1}+a_{i+2}+\ldots+2.a_{N-1}$
avec $n_0=1$ la relation (2) donne pour domaine de convergence : $|A_0|\leq 1$

Pour m=1, la relation (3) est vérifiée avec la séquence ($n_0\geq 0$, $n_{i+1} = n_i+1$)
car $a_i < 2.a_{i+1}$, car $a_i = \mathrm{atan}(2^{-n_i}) < 2.\mathrm{atan}(2^{-(n_i+1)})$, car $2.\mathrm{atan}(z) = \mathrm{atan}[2.z/(1-z^2)]$
d'où : $a_i < 2.a_{i+1} < a_{i+1}+2.a_{i+2} < \ldots < a_{i+1}+a_{i+2}+\ldots+2.a_{N-1}$
avec $n_0=0$ la relation (2) donne pour domaine de convergence : $|A_0|\leq 1.74$ ($K\approx 1.65$)

Pour m=-1, $\mathrm{atanh}(z) = z + z^3/3 + o(z^5)$ pour $|z|<1$,
donc pour z>0, $\mathrm{atanh}(z)-\mathrm{atanh}(z^3/2) < z < \mathrm{atanh}(z)$, donc pour $k \leq 3i+1$:

$$\mathrm{atanh}(2^{-i}) - \mathrm{atanh}(2^{-k}) < 2^{-i} < \mathrm{atanh}(2^{-(N-1)}) + \sum_{j=i+1}^{N-1} \mathrm{atanh}(2^{-j})$$

En conséquence, la relation (3) est vérifiée avec la séquence :
($n_0>0$, $n_{i+1}=n_i+1$), excepté ($n_i=p_j=n_{i+1}$) avec ($p_0\leq 3.n_0+1$, $p_{j+1}\leq 3.p_j+1$)
avec $n_0=1$, la relation (2) donne pour domaine de convergence : $|A_0| \leq 1.13$ ($K \approx 0.80$)

Réduction des facteurs radiaux

En généralisant le mécanisme de répétition nécessaire à la satisfaction du critère de convergence pour m=-1, il est facile de répéter certaines rotations élémentaires de façon à obtenir :

$$K = 2^m$$

Le critère de convergence reste satisfait, car les rotations élémentaires supplémentaires sont soustraites du membre gauche de l'inéquation (2), et le domaine de convergence s'en trouve étendu.

La table suivante donne le nombre de répétitions pour chaque différent n_i dans le cas de flottants de mantisse définie sur 24 bits (plus 1 bit de signe):

ni=	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	...25	
m=+1	1	2	3	3	4	2	3	2	4	2	3	4	1	1	1	1	1	... 1	N=47
m=0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	... 1	N=25
m=−1	0	4	3	2	2	2	2	2	3	3	4	1	4	2	1	1	1	... 1	N=46

Pour m=1 comme pour m=-1, le nombre total d'itérations est moins du double, soit un peu moins de l'équivalent d'un cycle linéaire supplémentaire (pour m=0, aucune répétition n'est nécessaire puisque K=1).

Dans le cas de Gatan, non concernée par le facteur radial K, ou même dans le cas où K peut être balancé par un 1/K initial ($x_0=1/K$ et $y_0=0$ donneraient après rotation sans répétition $x_N=G\sin(z_0)$ et $y_N=G\cos(z_0)$), cette méthode présente un surcoût équivalent à un cycle linéaire.

Christophe LAVARENNE, 12 rue du Docteur Vuillième, 92190 MEUDON, FRANCE. Tél: (33-1) 45071341

Dans le cas de la racine (et donc de l'arc-sinus et de la conversion coordonnées cartésiennes-pôlaires), le facteur K est inévitable, et son élimination par une division par K (ou une multiplication par 1/K) coûterait un cycle linéaire, soit un peu plus que la réduction par répétition.

Par contre, dans le cas des transformations sur les nombres complexes (rotation vectorielle ou conversion coordonnées pôlaires<->cartésiennes), le facteur K doit être éliminé pour x et pour y, ce qui coûterait deux cycles linéaires, soit l'équivalent d'un cycle linéaire de plus que pour la méthode par répétition.

Dans le cas d'une implantation matérielle de l'algorithme, la méthode par répétition l'emporterait pour son uniformité, les répétitions étant de toute manière inévitables pour satisfaire le critère de convergence hyperbolique.
Dans le cas de CORDICFP, qui est une implantation logicielle de l'algorithme, on a bien sûr choisi la méthode la moins coûteuse dans chaque cas.

IMPLANTATION

Largeur des registres et précision

Les valeurs chargées dans les registres (mantisses signées et normalisées des flottants à traiter) sont toujours dans l'intervalle [-1,-1/2[U [1/2,1[.
La précision de l'algorithme, définie par sa dernière itération grâce aux critères de convergence, doit être meilleure que le celle du bit le moins significatif des mantisses, soit, pour des nombres flottants de mantisses définies sur L bits (plus 1 bit de signe):

$$\partial(x_N) = |x_N - x_{N-1}| \leq |\max(y)|.2^{-n_{N-1}} \leq 2^{-n_{N-1}} < 2^{-L}$$
$$\partial(y_N) = |y_N - y_{N-1}| \leq |\max(x)|.2^{-n_{N-1}} \leq 2^{-n_{N-1}} < 2^{-L}$$
$$\partial(z_N) = |z_N - z_{N-1}| \leq a_{N-1} \approx 2^{-n_{N-1}} < 2^{-L}$$

soit au minimum : $n_{N-1} = L+1$

Comme K=2 pour m=1, les registres x et y doivent pouvoir contenir des valeurs dans l'intervalle [-2,-1/2[U [1/2,2[, ce qui convient également pour le registre z qui contient pour m=1 des valeurs au plus égales à π/2=1.57 (cf IDENTITES dans BASES MATHEMATIQUES).
En conséquence, les registres doivent avoir au moins un bit entier de garde.

Comme les registres sont de longeur finie, chaque accumulation (une par itération) induit une erreur de troncature égale à la moitié du bit le moins significatif, soit N/2 fois le bit le moins significatif pour N itérations. Comme N<2(1+L) (voir réduction du facteur radial), le nombre G de bits de garde des registres doit être au moins :

$G \geq \mathrm{Sup}[\log_2(1+L)]$ (Sup = valeur entière par excès)

et les constantes $\mathrm{Gatan}(2^{-n_i})$ doivent être précalculées avec (L+G) bits de précision, afin que la propagation des erreurs de troncature s'arrête avant le bit le moins significatif des mantisses.

Au total, le nombre LG de bits d'un registre gardé doit donc être au moins :

$LG \geq 2 + L + G$ (bit de signe, bit entier, bits significatifs, bits de garde)

Dans le cas d'une implantation matérielle, les registres et chemins de données auront LG bits.
Dans le cas de CORDICFP, chaque registre ou constante est stocké dans un nombre entier LM de mots mémoire de 16bits (d'adresse paire, la plupart des processeurs actuels privilégiant ce type d'adressage) :

$LM = \mathrm{Sup}(LG/16)$ (Sup = valeur entière par excès)

Génération des constantes

Pour m=0, le calcul des 2^{-i} est trivial, permettant une économie de stockage importante.
Dans le cas d'une implantation matérielle, la logique de décodage d'adresse pour N constantes génère N lignes d'adresses, dont une seule est à l'état haut, générant ainsi directement les 2^{-i}.
Dans le cas d'une implantation logicielle avec des mots mémoire de 16bits, le stockage contigü des 2^{-i} (matrice de bits diagonale) fait apparaître une périodicité T=(16*LM-1)mots ; en conséquence, il suffit d'une zone de stockage de 17*LM mots, et on trouvera la constante 2^{-i} à (i*LM)modulo(T) mots du début de cette zone ; ces adresses peuvent être précalculées.

Pour m=1 et m=-1, les Gatan(2^{-i}) sont calculées par leur développement en série limité :

$$\text{avec } i>0: \text{Gatan}(2^{-i}) = \sum_{j\geq 0}(-m)^j.2^{-i.(2j+1)}/(2j+1)$$

Pour chaque j, 1/(2j+1) est calculé en une vectorisation linéaire, et pour chaque i, le résultat est décalé i.(2j+1) fois avant d'être accumulé à la ième constante (avec un signe négatif pour m=1 et j impair). Les termes tels que i.(2j+1)≥(L+G) ne sont plus significatifs ; en particulier, pour i≥(L+G)/3, seuls le terme en j=0 est significatif, et comme ce terme vaut 2^{-i}, seules les (L+G)/3 premières constantes doivent être stockées, pour m=1 et pour m=-1, les autres pouvant être générées comme dans le cas linéaire.

Avec i=0 pour m=1, le développement limité converge encore, mais désespérement lentement; atan(2^{-0}) est plus rapidement obtenue par vectorisation trigonométrique avec $x_0=y_0=1$ et $z_0=0$ qui donne z_N = atan(x_0/y_0) = atan(1) = atan(2^{-0}); la série ($n_0=1$, $n_{i+1} = n_i+1$) suffit, car elle ne nécessite que les atan(2^{-i}) calculées précédemment pour i>0, et son domaine de convergence est 0.96 > atan(1) = $\pi/4$ = 0.79.

Le logarithme naturel de 2, nécessaire aux identités de changement d'échelle pour m=-1, s'obtient par vectorisation hyperbolique :

$$\log_e(2) = -\log_e(1/2) = 2.\text{atanh}[(1-1/2)/(1+1/2)]$$

Portabilité

La portabilité des données est assurée en suivant le standard IEEE, tant au niveau ASCII (notation exponentielle et format libre) qu'au niveau binaire (les nombres peuvent être stockés et récupérés aux formats mémoire IEEE 32 et 64 bits).

CORDICFP a été conçue et écrite en vue de sa portabilité sous Forth83 : développée à l'origine en NEON sur Macintosh, sa première version commerciale (distribuée par la société anglaise MPE) a été écrite en Modular-Forth sur IBM/PC et sur RTX2000, une seconde version (distribuée par l'association Jedi) a été portée en Turbo-Forth sur IBM/PC, et une version est en projet sur 68000 (probablement en Volks-Forth sur Atari); une version matérielle est également en cours d'étude.

L'efficacité de CORDICFP dépend essentiellement d'une dizaine de primitives (transferts de registres avec décalages ou accumulations). Une version Forth83 de ces primitives permet un portage immédiat sur différents processeurs, et donne le fil conducteur pour leur implantation en assembleur, pour une exécution globalement dix fois plus rapide environ.

Performances

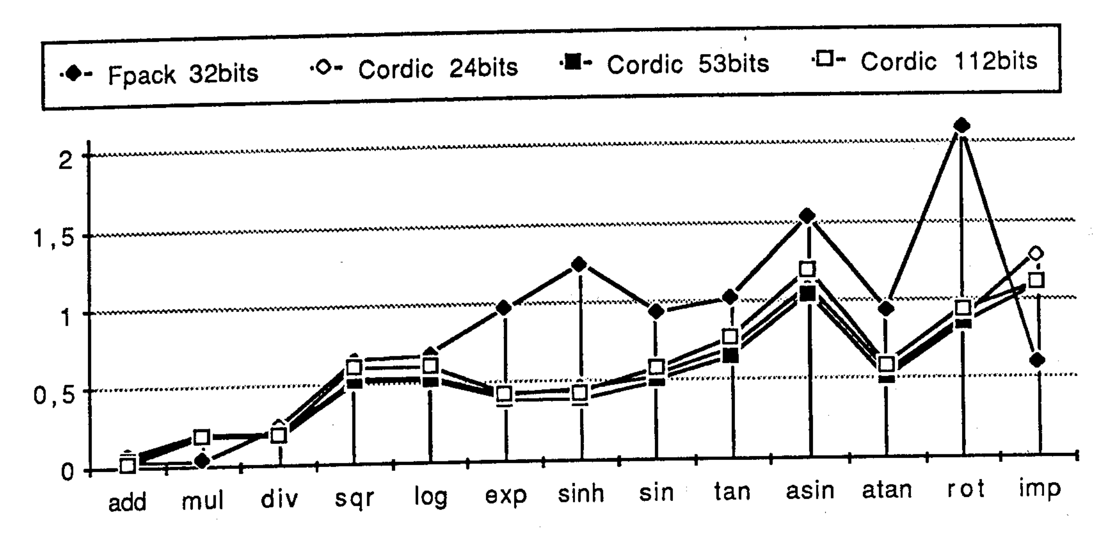
Les performances (millisecondes) donnés ci-dessous ont été mesurés sur un PC/AT CompaqII (processeur Intel 80286 à 8MHz) en Turbo-Forth avec la version assembleur des primitives, pour 24, 53 et 112 bits significatifs de mantisse. A titre de comparaison, les temps mesurés dans les mêmes conditions pour la bibliothèque F-PACK (approximations polynomiales, mantisses 32 bits) sont donnés entre parenthèses; le graphique permet une meilleure comparaison, en rapportant les temps au nombre de bits significatifs.

Christophe LAVARENNE, 12 rue du Docteur Vuillième, 92190 MEUDON, FRANCE. Tél: (33-1) 45071341

La multiplication de F-PACK est implantée avec UM* qui utilise le multiplieur du 80286 (13 cycles d'horloge pour un produit 16*16); CORDICFP produira une performance équivalente dans ses prochaines versions, au prix d'un codage particulier de la multiplication.
L'impression de F-PACK est également plus efficace, car elle porte sur une mantisse 32bits pour laquelle sont faites les routines de conversion ASCII qui utilisent le diviseur du 80286 (25 cycles d'horloge pour une division 32/16).
Partout ailleurs, CORDICFP est beaucoup plus efficace, surtout bien sûr pour les fonctions manipulant des nombres complexes, et montre une corrélation quasi-linéaire entre le temps d'exécution et le nombre de bits significatifs.

(32)	24	53	112	FONCTION
(1.2)	1.6	1.7	1.8	addition et soustraction
(1.2)	5	10	22	multiplication
(8)	5	10	22	division
(21)	13	28	69	racine carrée
(22)	13	28	69	logarithme népérien
(31)	10	21	49	exponentielle
(40)	11	21	49	sinus et cosinus hyperboliques
(30)	13	27	65	sinus et cosinus trigonométriques
(33)	17	35	87	tangente trigonométrique
(49)	26	55	134	arc-sinus et arc-cosinus trigonométriques
(30)	13	27	66	arc-tangente trigonométrique
(67)	21	45	105	conversions et rotation complexes
(19)	31	58	124	impression d'un nombre

Performance comparatives (ms/bit)



BIBLIOGRAPHIE

AHMED, 1981
Signal processing algorithms and architectures Ph.D, Elec Eng Dept., Stanford Univ, 1981
COCHRAN, 1972
Algorithms and accuracy in the HP-35, Hewlett-Packard journal, pp10-11, June 1972
HAVILAND-TUSZYNSKI, 1980
A cordic arithmetic processor chip, IEEE Transactions on Computers volC29 no2, Feb 1980
VOLDER, 1956
Binary computation algorithm for coordinate rotation and function generation
Convair Report IAR-1148 Aeroelectronics Group, June 1956
WALTHER, 1971
A unified algorithm for elementary functions, Spring Joint Computer Conf 1971, pp379-385

Christophe LAVARENNE, 12 rue du Docteur Vuillième, 92190 MEUDON, FRANCE. Tél: (33-1) 45071341

(suite de la page 8)

FORTH7, SINON CE N'EST PLUS DU JEU... IL NE M'EN VOUDRA PAS.)

-PICK REMET A UNE POSITION N DE LA PILE UNE VALEUR PREDEPOSEE SUR SON SOMMET:

```
VAL N ---
```

EXEMPLE: 12 15 18 DEPOSE 3 VALS SUR LA PILE:
10 1 -PICK . . .
AFFICHE 18 10 12, 10 AYANT REMPLACE 15.

IAPX 09.08.89 03h44
REPONSE AU CHALLENGE DE SECRETAIRE (05.08.89 23h41)

```
: -PICK ( val n -- )
1+ SWAP OVER DUP 1- >R >R >R ROLL DROP
R> R> R> 0 DO DUP >R ROLL R> LOOP
DROP ;
```

Cette solution n'est certes ni la plus rapide ni la plus compacte (on pourrait ecrire directement dans la pile operande par l'intermediaire du registre de pile) mais elle a au moins le merite d'etre hyper-portable, dans la mesure ou elle n'exploite pas les particularites intimes du FORTH sur lequel elle est implantee (raison pour laquelle, d'ailleurs, elle n'est ni la plus rapide, ni la plus compacte!).

(De plus, l'intimite de TF83 m'est inconnue: faut bien se trouver des excuses!!!).

And may the FARCE be with you !!! BYE.

SECRETAIRE 09.08.89 10h29
TRES BIEN, IAPX A GAGNE UN MODULE TF83 DE SON CHOIX M1 A 8... QU'IL ME RAPPELLE SES COORDONNEES DANS MA BAL, SVP.

A QUOI PEUT SERVIR -PICK? A FAIRE DES OPERATEURS NON DESTRUCTIFS A LA MANIERE DE C:

```
empilage A
empilage B
empilage 0 pour resultat
 --> APPEL PROCEDURE
     1 PICK 2 PICK + 0 -PICK
 <-- RETOUR DE PROCEDURE
exploitation parametre RESULTAT
depilage B
depilage A
```

VOICI LA PREMIERE PIERRE D'UN COMPILATEUR C POUR GENERER DU CODE FORTH.

SECRETAIRE 09.08.89 15h26
NOUVEAUX SOFTS EN TELECHARGEMENT:

- FORTH 1:
  DIGIT.FTH ET MOIRES.GRS POUR VISU D'IMAGES DIGITALISEES
- FORTH 4:
  HALT1.FTH, VERSION AMELIOREE DE HALT.FTH
  INOUT.FTH, UNE NOUVELLE STRUCTURE OUT..ENDOUT.
  DISASS86: DESASSEMBLEUR 8086 POUR TURBO. APRES COMPILATION, TAPER SEE mot ET SI mot EST DEFINI PAR CODE..ENDCODE, IL SERA DESASSEMBLE. FACILITE LA MISE AU POINT DES PROGRAMMES.
- FORTH 5: HANOI.FTH POUR JOUER AVEC LES TOURS DE HANOI
  CAPTURE.FTH, UN JEU STYLE PACMAN ASSEZ SIMPLISTE.
  BANNER.FTH POUR ECRIRE EN CARACTERES GEANTS SUR ECRAN.
  PREFUNI.FTH OU LA MANIERE DE DEFINIR DES CONSTANTES ET VARIABLES EN SERIE.
- FORTH 6: ASSMF32.FTH NOUVEAU TIRAGE DE L'ASSEMBLEUR F32, PAS FINI, MAIS DONT LA DOC SITUEE APRES EOF INDIQUE LA LISTE DES MNEMONIQUES EXPLOITABLES

VOUS ETES DEVELOPPEURS EN TURBO-FORTH, DESASS86 EST L'OUTIL QUI MANQUAIT A TURBO. TELECHARGEZ-LE, PUIS LANCEZ TURBO PUIS INCLUDE DESASS86. ENSUITE, SAUVEZ LE NOUVEAU TURBO EN TAPANT SUR LA TOUCHE F4 ET SUIVI DE LA FRAPPE DE TFAD (Turbo Forth Avec Desassembleur) ET BYE. MAINTENANT, LE NOUVEAU TURBO EST LANCE PAR TFAD DEPUIS DOS. A+

SECRETAIRE 09.08.89 15h38
LA NOTORIETE DE TURBO-FORTH NE CESSE DE S'ACCROITRE AU NIVEAU INTERNATIONAL: EN EFFET, UNE UNIVERSITE DE SILESIE, EN POLOGNE, VIENT D'EN COMMANDER UN EXEMPLAIRE.

TURBO-FORTH PASSE A L'EST. A QUAND UNE DEMANDE DE LENINGRAD OU MOSCOU... SI ELLE N'EMANE PAS DU KGB!

NOUS AVONS DEMANDE A FAIRE INSCRIRE LES COORDONNEES DE SAM*JEDI DANS L'ANNUAIRE DES SERVEURS FORTH PUBLIEE REGULIEREMENT PAR LA REVUE FORTH DIMENSION. ESPERONS QUE NOTRE DEMANDE SOIT PRISE EN CONSIDERATION.

FORTH7 09.08.89 20h26
PILES PRIVEES

-1-

Les programmes complexes font quelque fois de la pile un tel echeveau qu'on s'y perd entre les DUP SWAP et les -ROT 'n ROLL meme a lancer des -PICK ...

Si fragmenter le probleme en mots plus simples reste la base, il arrive qu'on se dise qu'une deuxieme pile aiderait a s'en tirer. Un exemple: ALEPH gere les variables locales avec une 2eme pile.

Une pile c'est: un espace memoire, un pointeur de pile, des operateurs E/S de de/em -pilage, et eventuellement des operateurs de pile.

1ere technique: definir la pile (CREATE STACK xx ALLOT) et toute sa mecanique (LIFO ou FIFO). Je n'y reviens pas: c'est decrit partout et bien transportable.

2eme technique: modifier le pointeur interne de la pile Forth. En TF-8088 on utilisera SP@ et SP! en sachant que la pile pousse vers le bas et qu'il est delicat de changer le segment de pile SS .
L'avantage est de recuperer pour cette 2eme pile toute la librairie de la pile usuelle.

```
VARIABLE SP     \ pointeur pile usuelle
VARIABLE SP2    \ pointeur 2eme pile

: S2   ( -- )  \ passe sur pile 2
  SP@ SP ! SP2 @ SP! ;
: -S2  ( -- )  \ retour pile usuelle
  SP@ SP2 ! SP @ SP! ;
```

Facile alors de definir des operateurs pour la pile privee:

```
: UDUP S2 DUP -S2 ;
```

Je passe vite sur l'empilage-depilage qui reste classique:

```
: UPUSH  -2 SP2 +! SP2 @ ! ;
: UPOP   SP2 @ @  2 SP2 +! ;
```

3eme technique (de plus en plus Forth): utiliser le multitache. Chaque nouvelle tache possede sa propre pile: les manipulations de SP que nous venons de voirs ont deja toutes pretes dans la gestion multitache de TF!

Le multitache n'est pas seulement fait pour des taches independantes (spooler ou horloge) d'arriere-plan mais peut se concevoir comme un remarquable outil de division du travail. Un gros probleme peut etre traite par plusieurs taches reliees entre elles, concues dans un esprit de traitement parallele.

J'ai longtemps cru que le multitache F83 etait un gadget ou un exercice de style du savoir-faire Forth (on s'est

trop arrete a BACKGROUND:). Il a meme ete question de l'exclure du noyau (mais il eut fallu virer aussi les USER devenues inutiles!) en arguant que le multitache 'serieux' serait desormais l'affaire des nouveaux DOS ...

La multiprogrammation est une des voies pour apprehender le gigantisme qui commence a nous cerner de toutes parts: developper le multitache FORTH est une PRIORITE: le langage s'y prete si bien.

FF32 10.08.89 21h21
DE MES ILES BRETONNES JE VOIS LE FORUM REPRENDRE VIE, BRAVO. DANS UNE DIZAINE JE RENTRE CHEZ MOI ET REPRENDRAI LE BATON DE PELERIN POUR F32. PAS DE REACTION SUR LE SIMULATEUR S32. COMMENT FAIRE? POUVEZ TOUJOURS ME PARLER, J'ECOUTE. POUR LES PARISIENS, PRENONS RDV POUR LA SEMAINE 4/AOUT.

GARY 11.08.89 00h32
GARY SMITH, LITTLE ROCK USA:
IF ANY OF YOUR ASSOCIATION MEMBERS HAVE ACCESS TO UsenetIT WOULD MAKE ELECTRONIC CONTACT WITH YOU MUCH EASIER. BOTH THE FIG CHAPTERS COORDINATOR, JACK WOEHR (JAX) AND MYSELF (GARS) ARE EASY TO CONTACT VIA Usenet. IF POSSIBLE PLEASE CONTACT ME AT ONE OF THE FOLLOWING:
!ames!chinet.chi.il.us!UUCP!gars
sun!cgl.ucsf.EDU!wet.us!UUCP!gars
well!gars@lll-winken.arpa
sun!portal!cup.portal.com.us!UUCP!gars

Best regards,

IAPX 11.08.89 02h48
L'idee de FORTH7 consistant a permettre l'utilisation de plusieurs piles , en evitant les (trop) classiques >R R> et autres R@ , est tout a fait interessante. Bravo, donc, FORTH7.

Mais puisque nous parlons de pile, notion quelque peu fondamentale en FORTH(!), j'ai egalement une suggestion a soumettre aux honorables participants de ce FORUM. Il s'agirait de joindre a tout element present dans la pile (operande) une valeur indiquant le type de cet element (valeur sauvegardable soit dans cette meme pile, soit dans une seconde pile). Ainsi tout element serait identifiable, ce qui permettrait a un mot donne de reagir differemmen selon le type des parametres presents dans la pile lors de son execution.

Exemple : le mot '+' serait definit ainsi:

si les elements sont des entiers (VAR) -> + classique
si les elements sont des pointeurs chaine -> addition chaine
si le premier element est un pointeur chaine et le second un entier -> conversion de l'entier en chaine et addition chaine (et vice-versa) ....

Il ne serait plus necessaire de distinguer '.' de TYPE, ! de C!, '+' de APPEND$, une adresse courte d'une adresse longue, un pointeur car d'un pointeur entier, chaine, double mot.

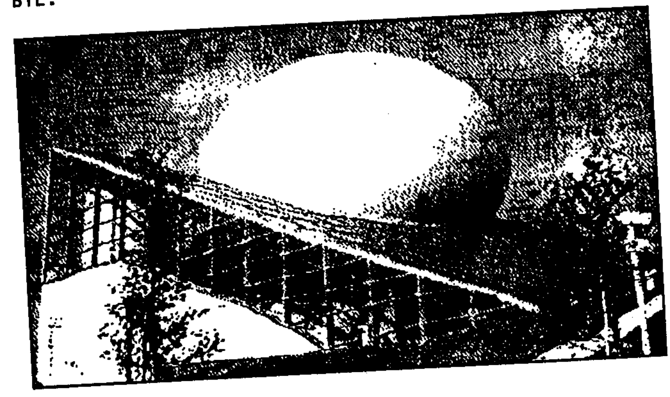
Ceci eviterait la multiplication de mots realisant la meme fonction mais portant un nom different car agissant sur des elements differents . Par exemple, DUP et 2DUP, DROP et 2DROP, C@ et @ (pourquoi pas 4DUP, 4DROP et Z@ ?) s'appeleraient simplement DUP, DROP et @.

De plus , ce systeme allegerait notablement le travail du programmeur , realisant d'une part des taches que le programmeur est aujourd'hui oblige de specifier, lui permettant d'autre part d'ecrire des routines polyvalentes qu'il ne serait pas necessaire d'ecrire en 10 exemplaires pour que leur mode d'execution soit flexible .

Cette idee s'oppose quelque peu aux FORTHs que nous utilisons, bien que ce ne soit pas mon propos (que les inconditionnels maitrisent leurs envies d'assassinat!). En effet, ces FORTHs ne connaissent jamais le type des elements sur lesquels les mots agissent, ce qui permet une grande souplesse. Mais rien n'empeche dansl'hypothese d'un FORTH connaissant le type des elements, de creer des routines de conversion de type.

Je precise que cette idee n'est pas mienne: elle a ete applique (un peu moins fondementalement que ce que je propose , mais de facon assez poussee pour en entrevoir les possibilites) sur la calculatrice de poche (pub) de la gamme28 de chez HP; son langage est tres proche du FORTH (en un peu moins permissif et plus specialise (scientifique)). Le resultat est tout a fait genial, bien que l'application de cette idee fasse quelque peu souffrir les performances (vitesse) de la machine . Enfin bon, veut-on un langage genial ou une machine rapide? A ceux qui veulent une machine rapide, je repondrais qu'ils n'ont qu'a programmer en assembleur (surtout pas en FORTH) en C ou en PASCAL: beaucoup de perdu. Aux autres je demande de reflechir a cette idee. Quant aux inconditionnels, j'accepterais volontiers leurs critiques censees et objectives .

Ayant ecrit un FORTH "classique", projettant d'ecrire (en FORTH bien sur) un FORTH tel que decrit ci-dessus, j'attends avec impatience toute suggestion.
BYE.



FUTUROSCOPE

SECRETAIRE 11.08.89 09h02
REPONSE A IAPX, je constate tout d'abord qu'il s'est couche bien "tot". Ceci dit, son intervention est tres pertinente. En effet, FORTH souffre un peu de cette proliferation de traitements divers infliges aux types de donnees extremement disparates: VARIABLES CONSTANT STRING 2VARIABLE etc...

Mais point n'est besoin de re-ecrire FORTH: il suffit de passer par la definition d'objets.

Je m'explique, mais en operant au prealable une petite digression par le C (ou PASCAL ou BASIC... au choix).

Pour exemple, le printf de C traite indifférement des chaines litterales, nombres litteraux, constantes, variables flottants, etc...

Mais si d'aventure vous definissiez un nouveau type, complex par exemple, il n'est pas possible de lui appliquer printf. Il faudra definir une fonction specialisee.

Si maintenant on définit 50 types de données avec pour chaque type au moins une demi-douzaine de fonctions de traitement, ceci nous fera au moins soit 300 fonctions differentes a gerer, que ce soit en C, PASCAL ou meme FORTH.

Je connais ce probleme de la proliferation des primitives, car en construisant mon programme de tele-saisie, j'ai affronte ce cas. Et j'ai resolu ce casse tete en appliquant tout simplement les principes deja diffuses dans RSINIT.FTH avec CREATE..METHODS>.

Exemple: soit trois termes generiques de traitement:
  GET pour saisir depuis le clavier
  STORE pour affecter depuis la pile
  PRINT pour afficher
et une collection de donnees:
  VALUE: pour des flottants,
  DATE: pour des dates, format JJ/MM/AA
  STRING: pour des chaines

Dans mon programme, on peut faire des operations du style:

```
  5 VALUE: tva
  : STORE-TVA  " 18.6" STORE TVA ;
  10 STRING: libelle
  : GET-LIBELLE
   GET LIBELLE ;
: traitement ( ---)
  store-tva  get-libelle
  cr print tva
  cr print libelle ;
```

PRINT s'applique aussi bien a TVA qu'a LIBELLE et peut s'appliquer a n'importe quel nouveau type de donnees declare par la suite.

Comment resoud-on le probleme des primitives? Partons simplement d'une disposition X Y des donnees et traitements:

	GET	STORE	PRINT
VALUE:	get-val	store-val	print-val
DATE:	get-date	store-date	print-date
STRING:	get-str	store-str	print-str

Avec trois traitements et trois type de donnees on a deja 9 primitives.

Dans la realite, mon programme de telesaisie ne comporte pas moins de 7 traitements et 6 type de donnes.

Ainsi, en ecrivant simplement:

```
 5 VALUE: tva
 10 STRING: libelle
 DATE: jour-depart
 : TEST ( ---)
 cr print tva
 cr print libelle
 cr print jour-depart ;
```

Puis en decompilant TEST par SEE TEST, on voit s'afficher:

```
 : TEST
 CR TVA PRINT-VAL
 CR LIBELLE PRINT-STR
 CR JOUR-DEPART PRINT-STR ;
```

Rajoutez a tout ceci la possibilite d'affecter a une VALUE: indifferement une chaine ou un flottant, le traitement de flottants en notation infixee, on obtient ceci:

```
FVARIABLE VAL1
FVARIABLE VAL2

 : TEST
  ....  " 18.6" STORE TVA
 F[ ( VAL1 F@ + VAL2 F@ ) F/ F' 2 ]F
 STORE VAL2 ... ;
```

nota: le + est l'operateur d'addition en flottant quand il est utilise entre F[et]F. Je ne peut encore me passer des F@. Pour ce faire, il faudrait definir un type FVAL en tant qu'objet.

Pour resumer, depuis que j'ai applique la notion de L.O.O. (Langage Oriente Objet) a la manipulation de mes donnes FORTH pour les nouveaux types, j'ai pu considerablement simplifier mes programmes et ameliorer sensiblement leur lisibilite.

Je souhaite et desire meme que l'on poursuive plus avant le developpement des outils L.O.O. pour FORTH, car ça me semble plein de promesses.

Sans refondre FORTH, on reservera l'usage des primitives au developpement de la couche L.O.O. comme ailleurs on utilise l'assembleur.

SECRETAIRE 11.08.89 14h01
ANIMATION FIL DE FER 3D TEMPS REEL: GRACE AUX NOUVEAUX PROGRAMME 3DJ.FTH DJ.FTH 3DJA.FTH ET LINA.FTH QUI ANIMENT EN PERSPECTIVE 3D UN OBJET A L'ECRAN. FONCTIONNE SUR SYSTEME EQUIPE VGA OU EMULATION VGA. ADAPTABLE AUX AUTRES SYSTEMES. DEMO EPOUSTOUFLANTE SUR MON COMPAQ 80286

FORTH7 11.08.89 21h50
OBJECTIF OBJETS: C et Pascal sont en train de demontrer qu'un langage peut se mettre aux OBJETS sans perdre sa specificite propre. Le Forth-Objet existe deja a des degres divers: notre Secretaire fait la preuve qu'une implementation FOO tres simple peut etre tres efficace.

Mais nous sommes encore en pleine mouvance: les LOO introduisent une foule de nouveaux concepts: instanciations, heritages, methodes, messages, classes, sous-classes et metaclasses, surcharge, encapsulation, parallelisme, early and late binding...

Tous ces concepts sont bien developpes en Forth mais nous n'avons pas encore de produit fini. Remarquez d'ailleurs les memes hesitations en C Pascal ADA et les autres sur la question "Jusqu'ou porter les objets?".

Je n'ose vraiment pas proposer la constitution d'un nouveau groupe de travail sur les objectifs du Forth-Objet mais si vous avez des idees...

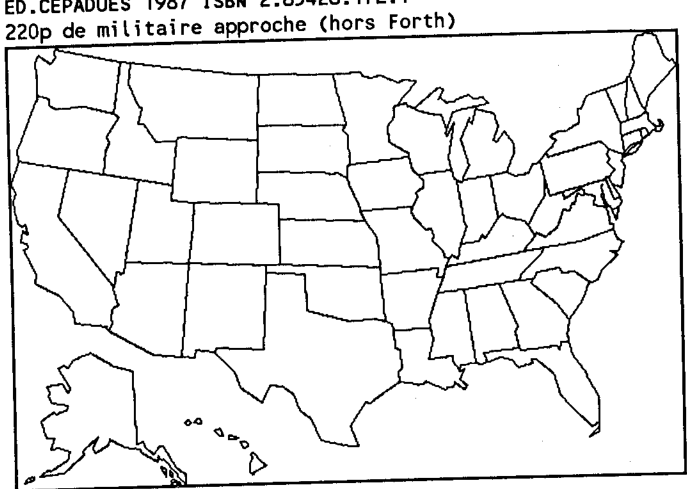
Et si vous n'en n'avez pas, rendez-vous les 24-26 Nov 89 a Monterey au bord du Pacific, au congres FORML sur ce sujet!

OBJECTIVEMENT: si comme moi votre patron vous refuse le voyage en Californie sous pretexte que le surf est trop dangereux en cette saison, voici un peu de lecture:

OBJECT-ORIENTED FORTH
Implementation of Data Structures
par Dick Pountain
ACADEMIC PRESS 1987 ISBN 0.12.563570.2

116 pages de base trop courtes mais indispensables: records, abstract data, types, lists, memory management using a Heap...

LES LANGAGES ORIENTES OBJETS /Cl.Bailly
ED.CEPADUES 1987 ISBN 2.85428.172.1
220p de militaire approche (hors Forth)



FORTH7 15.08.89 12h42
Quand on fait 'TYPE TEXTE.DOC' sur un fichier WordPerfect (ou autre TdTexte) c'est illisible. Rageant de devoir charger WP seulement pour voir ses fiches! Reprenant FILTER, me suis compile vite fait un filtre WPX.COM qui fait le menage des codes WP et me permet de taper 'TYPE TEXTE.DOC | WPX' ou 'TYPE TEXTE.DOC | WPX | MORE'... L'etait bien foutu MS-DOS quand meme!

Quand je lis dans les magazines US les sources en C de tels petits utilitaires je - me - fends - la - poire.

FORTH7 19.08.89 14h46
MULTIDEFINITIONS: dans le n.52 de JEDI et telechargement, Ch MOHR offre avec PREFUNI une methode de multidefinition de types de donnees.

Si les litanies de VARIABLE ou CONSTANT ne sont pas vraiment genantes en Forth quand elles sont bien presentees, il faut considerer la multidefinition plus comme un jeu d'analyse syntaxique.

Il est possible d'etendre l'interpreteur Forth pour installer un autre systeme de multidefinition generalise a tout type de donnee, multiligne, simple et sans containte.

EXTENSIONS DE L'INTERPRETEUR FORTH. L'interprete Forth en execution est:
1) je prends un mot dans le flot source
2) si je le connais, je l'execute
3) si c'est un nombre, je l'empile
4) sinon je signale une erreur

Il se trouve que l'etape 4 est vectorisee dans 3 justement pour permettre des extensions a l'interpreteur.

Nous pouvons remplacer l'etape 4 par 4) je considere ce mot comme nouveau et l'ajoute dans ma base de connaissance (technique cognitive classique en I.A.)

DEFINITIONS GROUPEES: appliquons ce principe d'extension a la multidefinition pour creer le mot GROUP tel que:

GROUP <datatype>

obligera l'interpreteur a considerer tout nouveau mot comme etant a definir dans le type de donnee <datatype>.

Un mot END-GROUP rendra a l'interprete son fonctionnement normal.

Voici sur l'ecran suivant un exemple de ce que donnerait l'utilisation de GROUP pour diverses donnees. Chacune utilise sa propre syntaxe de definition.

```
\ EXEMPLE DE DEFINITIONS GROUPEES
GROUP VARIABLE    AMELIE  BENOIT  CLAUDE
                  DENISE  EUGENE  FABIEN

GROUP CONSTANT    32 CARTES \ sans joker
                  50 PIONS \ commentaire

GROUP STRING      30 ANNONCE$
                  80 ENJEUX$

GROUP WINDOW      20 20 TABLE
                  40 10 MARQUE

GROUP DEFER       ACTION  REFLEXION  JEU
END-GROUP
```

DYNAMIQUE DE GROUP...: cette presentation groupee des donnees en jette a defaut d'etre indispensable!

Les structures meme non standards sont groupables et on peut mixer des commentaires ou d'autres commandes.

L'ecran suivant (seul) vous propose une definition de GROUP (F83 et TF).

Le mot est securise en execution: il est bien-sur impensable de creer un mot en pleine compilation d'un autre!

Votre chevalier JEDI, F7

```
DEFER DATATYPE     \ vecteur definisseur

: BACKWORD    \ recul d'un mot dans flot
  >IN @ 'WORD C@ - 1- 0 MAX >IN ! ;

: END-GROUP  ( -- )
  ['] (NUMBER) IS NUMBER ;

: (GROUP)     \ definisseur implicite
  NUMBER? NOT
  IF  STATE @
      IF END-GROUP TRUE ?MISSING THEN
 2DROP DPL OFF BACKWORD DATATYPE THEN ;

: GROUP  ( <mot> -- )
   '  IS DATATYPE
  ['] (GROUP) IS NUMBER ;
```

FORTH7 25.08.89 20h49
WRITELN(' Le silence eternel de ces')
WRITELN(' espaces infinis m'effraie')
WRITELN(' Blaise PASCAL')

Quand le Forum est ainsi desert, je me console avec le PASCALISSIME: depuis qu'il est en kiosque, je ne le manque pas. Equipe sympa, bons sujets, bonnes idees: un regal.

Dommage que les disquettes d'accompagnement ne soient pas PLUS cheres: on n'economise que 30 Frs a taper leurs programmes en... FORTH. Une vraie misere ...

SURRELYVES 28.08.89 22h13
JEDI 52: PAGE 4 DANS LE COMMENTAIRE OUT...ENDOUT IL MANQUE L'AROBAS @ APRES ENDOUT DANS L'EXEMPLE.

DANS LE COMMENTAIRE DE IN...ENDIN, IL FAUT LIRE: NUMEROTEES DE 0 A N-1 BIEN SUR...

PAGE 16: LE MOT EXEC: CORRESPOND AU MEME CONCEPT QUE OUT...ENDOUT COINCIDENCE!

FORTH7 31.08.89 14h25
TABLES DE PROCEDURES
OUT EXEC: tout comme CASE: ou les CREATE table] <proc>..<proc> [sont diverses conceptions de tables de procedures en Forth.

Chacune repond a un besoin propre avec son propre champ d'application:
- nombre de vecteurs predefinis ou non
- table externe (compilee a part)
- table interne (dans une definition)
- procedures simples (un seul mot) ou sequentielles
- controle ou non des limites de table
- aptitude ou non a la revectorisation
- premier indice 0 ou 1 ... etc

Avec tout langage defini, lorsqu'on a besoin d'un modele complexe, il faut soit 'faire avec' (ON GOSUB) soit bidouiller (GOTO 1000+100*N)

Dans tous les cas le genie du programmeur consiste a adapter son projet aux outils dont il dispose.

La demarche en Forth est generalement inverse: adapter l'outil au projet.

Si on reprend EXEC: dont on admirera la simplicite:

```
: EXEC: 2* R> + PERFORM ;
```

on peut etre sur qu'il ne conviendra pas a toutes les situations car n EXEC: force un EXIT (r> drop) sur l'execution du n+1 mot qui suit EXEC: sans compter que n debute a zero ce qui n'est pas tres 'intuitif' et que les risques de plantage sont colossaux si n sort de la table EXEC:

Et pourtant EXEC: est l'outil parfait du desassembleur. Tout autre eut ete trop lourd ou inadapte.

Mais personne ne vous empechera de reprendre le principe d'EXEC: pour que la table puisse s'integrer dans une definition, debuter a l'indice 1, et verifier le domaine de celui-ci. En voici un exemple:

```
\ OF:       ( n nmax --- )
\ execute le n-ieme parmi les nmax mots qui suivent

: OF:       ( n nmax --- )
  R> 2DUP SWAP 2* + >R -ROT MIN 1 MAX
  2* + 2 - PERFORM ;

: UN     ." UN " ;
: DEUX   ." DEUX " ;
: TROIS  ." TROIS " ;

: ESSAI    ( n --- )
  3 OF: UN DEUX TROIS ." TOUR(S)" ;

1 ESSAI
2 ESSAI
3 ESSAI
```

Remarquez ceci: en voulant 'ameliorer' le comportement de EXEC: j'ai ecrit le le mot OF: dont le moins qu'on puisse dire est qu'il est L A I D :

- la definition est tant alambiquee qu'il est devenu impossible de la comprendre a sa seule lecture
- le mot est devenu ambigu: deux arguments sur la pile, l'un libre, le deuxieme devant etre pre-compile avant lui.

La notation (n nmax --) devient trompeuse (dangereuse).

C'est un OUTIL -moche-. L'ideal serait de n'en conserver que sa FONCTION...

SECRETAIRE 04.09.89 16h50
JE SUIS REVENU APRES 3 SEMAINES D'ABSENCE. BEAUCOUP DE COURRIER DANS MA BAL ET A PART FORTH7, PRESQUE PERSONNE NE S'EST EXPRIME DANS LE FORUM. LE SOLEIL A-T-IL DONC TAPE SI DUR QUE CA?

ENCORE ET TOUJOURS: SI VOUS AVEZ DES PROBLEMES AVEC F83, TURBO-FORTH ET MAINTENANT F-PC, LE FORUM EST LA HOT-LINE (UN PEU DE FRANGLAIS POUR FAIRE GRINCER LES DENTS AUX FRANCOPHILES PURS ET DURS) DU FORTH.

IDEM SI VOUS POTASSEZ F83 CP/M SUR AMSTRAD (OU SCHNEIDER POUR LES ACHETEURS DE MATERIEL MADE IN GERMANY), OU VOLKS-FORTH SU ATARI.

UN TURBO-TURBO: UNE NOUVELLE VERSION F83 A FAILLI S'APPELER F88; ELLE PORTE LE NOM DE F-PC ET NOUS VIENT DES USA.

ELLE TIENT SUR AU MOINS 4 DISQUETTES ET EST BOURREE DE FICHIERS COMPRESSES. AU TOTAL PAS LOIN DE 3 MO DE CODE UNE FOIS DECOMPRESSE LE CONTENU DES FICHIERS.

F-PC TRAITE COMME TURBO-FORTH DES FICHIERS ASCII (EXTENSION .SEQ) DONT CERTAINS SONT COMPATIBLES AVEC TURBO.

IL COMPILE TRES VITE, EST DOCUMENTE EN ANGLAIS, DISPOSE DE NOMBREUX FICHIERS D'EXEMPLE.

RESTRICTIONS: NECESSITE AU MINIMUM 380 KO MEM, DEUX DRIVES OU MIEUX, UN DISQUE DUR (QUASI-OBLIGATOIRE). IL OCCUPE PRATIQUEMENT TOUTE LA MEMOIRE.

F-PC EST UN FICHIER .EXE MAIS NE TRAVAILLE PAS EN 32 BITS. LE CODE ACTIF EST TOUJOURS LIMITE AUX FATIDIQUES 64K MEM CECI POUR NE PAS CONTRARIER SES PERFORMANCES. LES EN-TETES SONT SEPARES AINSI QUE L'EDITEUR CE QUI LAISSE QUAND MEME UNE MARGE DE TRAVAIL CONFORTABLE.

ET COMMENT CE PROCURER CETTE NOUVEAUTE?
EN NOUS ENVOYANT 130 FR (CHEQUE OU TIMBRES) POUR LES 4 DISQUETTES A:
ASSOCIATION JEDI
17 RUE DE LA LANCETTE 75012 PARIS

NOTA: F-PC EST DU DOMAINE PUBLIC ET PEUT DONC ETRE DUPLIQUE ET DIFFUSE SANS NOTRE AUTORISATION.

2EME NOTA: LA VERSION F-PC EST DECONSEILLEE AUX DEBUTANTS EN FORTH POUR UNE PREMIERE PRISE EN MAIN. IL FOURMILLE DE ROUTINES NON STANDARD.

3EME NOTA: C'EST AVEC SURPRISE QUE NOUS AVONS CONSTATE FORTH7 ET MOI-MEME QUE F-PC UTILISE QUASIMENT LES MEMES TERMES QUE TURBO-FORTH POUR CERTAINES ROUTINES. ANSI, UN FICHIER .SEQ EST COMPILE EN TAPANT:

```
INCLUDE fichier.SEQ
```

A+

FF32 05.09.89 12h57
SILENCE DE MORT AUTOUR DE S32...

FORTH7 05.09.89 14h54
SIMULATEUR: appareil capable de reproduire le comportement d'un appareil.

A mon avis, le 'blocage' S32 tient dans un paradoxe: appareil simulant l'appareil qui devra le simuler.

J'exagere bien-sur! Mais songez un peu a ceci: le FORTH est d'abord une MACHINE A PILE d'un modele tres simple (quelques registres et une logique quasi-simpliste)

Nous desirons TOUS que le Forth soit une machine REELLE: certains estiment que c'est le seul avenir pour FORTH.

La simplicite du FORTH autorise a rever d'une machine REELLE (elle existe deja) alors que nous travaillons encore sur des machines VIRTUELLES.

Ainsi Turbo-Forth-83 est avant tout un SIMULATEUR Forth sur IBM-PC!

Ce Forth est d'architecture classique c'est-a-dire a mots de 16 bits. Tout le monde est d'accord en 1989 pour estimer que la machine FORTH doit muter aux 32 bits. Definir cette nouvelle machine Forth est un jeu d'enfant: toutes les entites depuis les cellules de pile jusqu'aux pointeurs cfa sont 32 bits! Simple oui: VIRTUELLEMENT simple.

Pourquoi F32 c-a-d le SIMULATEUR de la machine virtuelle Forth-32 bits n'est pas operationnel?

Parce que nous developpons sur des machines 16 bits! Simuler par exemple une pile 32 bits est terriblement penalisant sur 8088. C'est possible bien-sur mais qui en voudra en pratique?

Et le S32? Si j'ai bien compris, c'est la couche SOUS le F32. C'est le simulateur au niveau le plus bas (les registres, les portes, l'ALU, les piles, le hard quoi) d'une machine REELLE F32. Et c'est la que tout le monde se met a DISJONCTER!

Disjoncter est bien le mot: personne ne lit les choses de la meme facon! Entre les machines reelles ou virtuelles qui se simulent les unes les autres, il y a de quoi se perdre...

S32 simule une machine electronique avec des transistors, des impulsions et tout ca (je n'y connais rien!) qui une fois cablee simule elle-meme une machine Forth 32 bits F32 virtuelle laquelle devenue enfin reelle sous la forme d'une puce encapsulee est a meme de simuler reellement S32. Ou est le probleme?

GUILLAUMAUD PH. 08.09.89 08h30
COMMUNIQUE: LE 14 SEPTEMBRE 1989, Presentation de la version 3.00 de Le-Forth

Quelques caracteristiques (en vrac):
- Standard 83
- Generation de fichiers ".EXE"
- Inclusion d'outils permettant l'acces a TOUT le systeme PC.
- Gestion de la memoire etendue (EMS)
- Graphisme (MGA,CGA,EGA,VGA)

En cours d'etude : Version specifique au 80286,80386,68000
A bon entendeur...

FORTH7 08.09.89 19h49
OUAOUH LE FORTH! Sommes bien contents d'apprendre le prochain lancement de la V3.00. Nous avions essaye la V1.05 il y a qq mois et notre Secretaire avait resume notre opinion: un produit tres prometteur par quelques innovations mais plutot un heritier du FIG ou du F79. C'est dire si la mise au niveau F83 etait attendue avec impatience!

Nous osons esperer que LE FORTH aura su integrer quelques unes des fonctionnalites de notre Turbo-Forth-83... Un Forth professionnel francais?

SURRELYVES 08.09.89 20h26
BRAVO A LE-FORTH

QUESTION: QUI LE DEVELOPPE? PRESENTATION OU CA? DANS LA PRESSE? ICI?

QUESTION PLUS EMBARASSANTE: Y AURA-T-IL UNE D O C U M E N T A T I O N ? FAUDRA-T-IL EPLUCHER UN SOURCE VAGUEMENT COMMENTE POUR SAVOIR L'OPERATION -CERTES GENIALE- EFFECTUEE PAR LE MOT NON STANDARD <*[FLOMP ?

LE FORTH NE SE DEVELOPPERA PAS S'IL N'EXISTE PAS UNE DOCUMENTATION SERIEUSE ET PEDAGOGIQUE EN LANGUE FRANCAISE. IL FAUT SOULIGNER ICI L'EFFICACITE DU SECRETAIRE DE JEDI QUI A DES JOURNEES DE 30 HEURES POUR TOUT FAIRE. NEANMOINS, LA DOC DISPONIBLE RESTE NOTOIREMENT INSUFFISANTE. A PAR CA, COMPLETEMENT LARGUE PAR F32 C'EST DONC SUREMENT GENIAL.

FORTH7 09.09.89 15h33
D O C U M E N T A T I O N !

Eternel serpent de mer: monstre proteiforme que nul ne voit...

Une vraie plaie: mille fois plus simple et gratifiant de developper que de documenter. Pas seulement en Forth mais particulierement en Forth.

La charge est telle qu'il faudra tot ou tard legiferer (encore que Forth soit le plus hors-la-loi des langages). Deux solutions radicales sont a envisager:

- documentation inutile
- documentation obligatoire

DOCUMENTATION INUTILE: Toutes les docs sont nulles: ca debute par 50 pages sur ce qu'est un ordinateur et ca finit par la table ASCII. Par definition on n'y trouve jamais ce qu'on y cherche.

Admettons le principe: vous lisez une bonne fois un bouquin sur la becane ou le logiciel, histoire de savoir ce qu'est un clavier ou une pile de RPN et puis termine! Surtout ne pas avoir a y chercher quelque chose plus tard: elle ne s'y trouvera pas. Ce sera au logiciel de vous repondre.

Je trouve que nous avons fait pas mal de chemin dans ce sens:

- decompilateur
- debogueur
- desassembleur
- auto-documentation
- presence de tous les sources

D'importants progres restent a faire mais vous comprenez ce que j'appelle la doc inutile: vous lisez le 'Leo BRODIE' vous lancez TF90 et vous vous dem..dez pour programmer en Forth... On y viendra.

DOCUMENTATION OBLIGATOIRE: restent les bibliotheques. C'est-a-dire le code mis au point par un et destine a etre re-utilise par tous. La, il va falloir sevir: doc obligatoire!

Reprenons l'idee de la doc inutile et supposons le systeme equipe deja d'une infrastructure d'autodocumentation complete. En gros: tout objet du systeme est lie a une base de donnee documentaire multilangue.

Et distinguons les applications (LOAD) des bibliotheques (INCLUDE) pour que celles-ci soient documentees:

Une application (le plus souvent fermee compilee sans en-tetes) pourra encore n'etre lisible que par son auteur, en toute liberte et sans charge de documentation.

Par contre, le meme programme pour etre promu au rang d'UNIT sera controle par le compilateur: formatage standardise du source, presence et linkage des informations documentaires. Ainsi on va forcer le developpeur a documenter en parallele. C'est la seule solution. En contrepartie l'UNIT pourra etre chargee intantanement sous forme pre-compilee.

DOCUMENTATION PAPIER: ce projet documentaire (larvaire...) devrait faire partie du cahier des charges du prochain langage car nous assistons a ceci: les langages ont besoin de plus de puissance, de plus de concepts et quand la doc est enfin prete le produit est obsolete.

La doc doit etre integree des la conception et disposer

elle-aussi des nouveaux concepts (hypertextes). Quant a la doc papier (plus-value protection ou tape-a-l'oeil) il ne sera pas bien sorcier de la pondre au kilo si le contenu redactionnel existe deja!

FF32 09.09.89 22h17
F32: NI DEBILE NI GENIAL; SIMPLEMENT UN TRAVAIL D'EQUIPE. JE NE SUIS QU'UN FAISEUR DE CHIP, MAIS QUI VOUS PREPARE UNE MACHINE FORTH REELLE ARCHI-BALAISE. CEPENDANT, JE NE PEUX RIEN SANS VOTRE SOUTIEN... QUI VA S'EN SERVIR SINON VOUS?

ALORS CELA ME PEINE UN PEU QUAND JE LIS DU SUPERLATIF SUR F32 DANS LE FORUM ET QUE MA BAL RESTE DESESPEREMENT VIDE.

QUOI DE PLUS SIMPLE QU'UN CONTACT? JE SUIS L'INTERLOCUTEUR LE PLUS GENTIL DU MONDE, ENCORE FAUT-IL QU'IL Y AIT QUELQU'UN EN FACE. SI MES PAPIERS SONT INCOMPREHENSIBLES, C'EST SUREMENT MA FAUTE. MAIS ALORS DITES-MOI CE QUE VOUS VOULEZ/POUVEZ/FAITES. VENEZ ME VOIR ET JE VOUS MONTREREZ COMMENT MARCHENT LES SYSTEMES MODERNES QUE JE TENTE DE VOUS PROPOSER. DONNEZ-MOI RENDEZ-VOUS ET JE VIENDRAI VOUS VOIR. SI QUELQU'UN VOIT CE QU'IL FAUT DIRE, IL LE DIRA POUR MOI

JE VOUS VOIS QUAND? IL Y A POURTANT TELLEMENT A FAIRE EN COUPLANT CE LANGAGE AVEC UNE MACHINE APPROPRIEE, ON CROIT REVER. SUIS-JE DONC LE SEUL, MOI QUI NE SUIS PAS FORTHIEN POUR DEUX RONDS, A REVER DANS JEDI...

SURRELYVES 11.09.89 21h05
1000 EXCUSES A F32 SI J'AI PU LE VEXER EN DISANT QUE J'ETAIS LARGUE PAR S32

N'ETANT NI DEVELOPPEUR DE CHIP NI INFORMATICIEN THEORICIEN, LES NOTIONS D'ARCHITECTURE INFORMATIQUE NE M'INTERESSE QUE MEDIOCREMENT

LA QUESTION QUE JE ME POSE EST: F32 (OU S32???) SERA-T-IL A LA FIN UN O U T I L UTILISABLE PAR UN FORTHIEN LAMBDA EQUIPE D'UN PC? A UN PRIX RAISONNABLE? SERA-CE UN MOTEUR DE FERRARI AU PRIX D'UNE FERRARI?

VOILA CE QUI BLOQUE PEUT-ETRE CERTAINS POUR S'INTERESSER DE PLUS PRES AU CHER ENFANT...

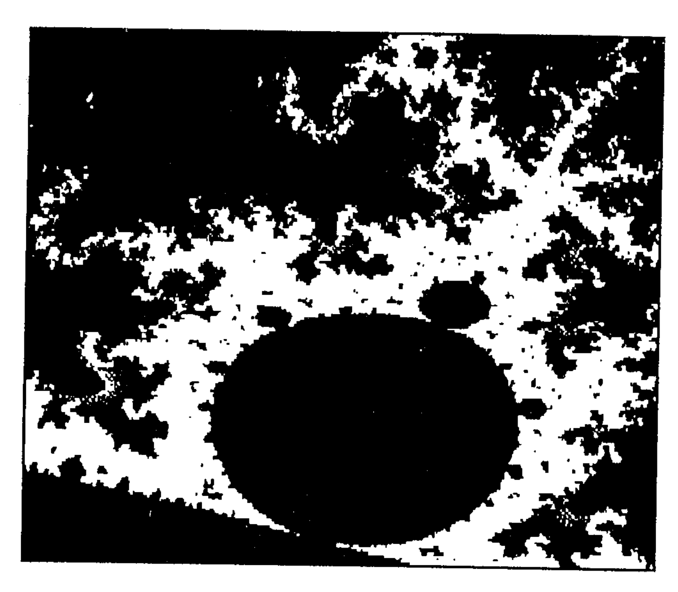
FORTHFOUR 13.09.89 21h39
BONJOUR A TOUS. MES VACANCES ETAIENT BONNES.... MERCI
JE VIENS DE LIRE LE FORUM D'AOUT, NON DESOLE, CA VOLE TROP HAUT, JE FAIT SIMPLEMENT PARTIE DES GENS QUI ONT BESOIN DE LIRE 3 FOIS LEUR DOC POUR COMPENDRE UN TEXTE SIMPLE, ALORS SANS DOC!

N'EMPECHE, JE NE SAIS TOUJOURS PAS COMMENT COMPILER LE NOM D'UN FICHIER POUR POUVOIR L'UTILISER SANS LE DEMANDER A L'UTILISATEUR. AU SECOURS. AMITIES

FORTHFOUR 15.09.89 20h54
BJR. BON J'AI TROUVE POUR UN FICHIER
" OPEN TOTO.FTH" $EXECUTE
MEME QUESTION POUR PLUSIEURS FICHIERS SIMULTANES. MERCI

FF32 16.09.89 17h56
A SURRELYVES: F32 DESIGNE UN MICROPROCESSEUR DONT LE DESIGN SE FERA AU COURS DE L'ANNEE SCOLAIRE QUI COMMENCE. LES IMPLEMENTEURS SONT DES ETUDIANTS DE 3EME CYCLE DE L'ESIEE (Ndlr: Ecole Superieur D'Informatique, Electronique et Electrotechnique; dispose d'une "Salle blanche"...) A NOISY-LE-GRAND (re-Ndlr: à 300m du domicile du Secrétaire JEDI).

JE GUIDE CE PROJET AVEC POUR OBJECTIF DEJA QU'IL ABOUTISSE, ET CE N'EST PAS SIMPLE. MA MOTIVATION: VOIR JUSQU'OU ON PEUT MONTER EN PUISSANCE AVEC LE COUPLE MACHINE DE PILE/FORTH. EN 32 BITS. POUR OPTIMISER L'UTILISATION DU SILICIUM IL NOUS FAUT METTRE ASSEZ RAPIDEMENT DANS LES MAINS DES GENS DE JEDI L'OUTIL QUI LEUR PERMETTRA DE JOUER AVEC UN MODELE COMPORTEMENTAL DE F32. S'AGIT PAS DE TRANSCRIRE VOS APPLICATIONS. AU PLUS QUELQUES DIZAINES D'INSTRUCTIONS, POUR S'ASSURER DE LEUR BON ENCHAINEMENT. C'EST S32. J'AI DECRIT DEJA CE QUE DEVRAIT ETRE CE MONITEUR. IL N'EST DONC MALHEUREUSEMENT PAS TELECHARGEABLE POUR L'INSTANT. MAIS BIENTOT, QUI SAIT?



IL EXISTE UN ASSEMBLEUR INITIAL PROPOSE PAR NOTRE SECRETAIRE (MERCI). IL Y A QUELQUES DIFFICULTES AVEC LA REPRESENTATION DES NOMBRES, QUI EST PLUTOT CELLE DU 68020 ET QUI PASSE MAL PAR UN PC; NEAMMOINS CE N'EST QUE DU DETAIL A CE NIVEAU DU PROJET, ET S'IL Y A PB DE CE COTE, IL DEVRAIT PLUTOT ETRE CONTOURNE POUR L'INSTANT. CE QUI COMPTE C'EST DE DONNER UN MONITEUR PERMETTANT A TOUS DE SENTIR FONCTIONNER F32.

CE QU'IL FAUT 'SENTIR':

- ACTIVATION PARALLELE DES PILES EN HARD INVISIBLE DU SOFT
- FETCH INSTRUCTIONS AVANCE
- PILE D'INSTRUCTIONS INTERNES
- IMPORTANT PARALLELISME INTERNE
- STRATEGIES NECESSAIRES POUR JOUER AVEC LA TAILLE VARIABLE DES INSTRUCTION (F32 N'EST PAS UN RISC), LEUR TEMPS D'EXECUTION DEPENDANT DE L'ETAT DES FETCH, COMME SUR DES '286, '386, '680X0, ETC.
- STRATEGIE DE RELOCATION DU CODE, F32 ETANT SENSIBLE AUX TEMPS D'ACCES (ICI AUSSI IL NE SE CONDUIT PAS EN RISC).
- EXECUTION D'UN NOYAU A BASE FLOTTANTE

SANS S32 ON PEUT FAIRE JUSTE UN F32 PIFOMETRIQUE, ET DEUX A TROIS FOIS RAPIDE QU'AVEC. LES POSSESSEURS D'UN 68000 SONT MIEUX PLACES POUR ACCEDER A DES SIMU PLUS AMBITIEUSES, MAIS CE SERA PLUTOT POUR LE PRINTEMPS

APPELEZ MOI DOM 69 20 45 90 POUR + ON RECHERCHE DES PARTICIPANTS. PAUL ORTAIS

SECRETAIRE 18.09.89 08h45
CONCERNANT SIMULATEUR F32: C'est presque le serpent de mer. Le simulateur premier jet envisage fonctionne un peu comme DEBUG de DOS; il digere du code F32. Mais comme l'assembleur F32 n'est pas encore fiabilise, on ne peut etre certain, en cas de probleme, de dire si cela provient de l'assembleur, du simulateur ou du code lui-meme.

1) l'assembleur: il depend des instructions disponibles sur le processeur F32. J'ai deja transmis a FF32 et FORTH7 les specifications d'un premier jeu d'instructions style "GUIDE DU Z80 de Rodnay Zaks". Ces specifications permettent seulement d'attaquer le probleme et non de figer ce jeu d'instructions. Si vous telechargez ASSMF32.FTH fourni sur ce serveur (section FORTH, chapitre HARRIS NOVIX et F32...) vous aurez